ianuarie 2000
Vol. 10/1

Lei 15.000

gazeta de informatica

Revista viitorilor specialiști in tehnologia informației

din sumarul numărului:

Grafica Turtle și Aplicații in Pascal

BEGIN

2000

CONCURSURI

Rezultate Runda 2

Clasament Rundele 1+2
Rezolvări Runda 2
Propuse Runda 4

SERIAL

Teoria grafurilor

• CEOI '99
Probleme rezolvate

ȘTIRI
despre ultimele procesoare

GAZETA DE INFORMATICĂ

PREMIUL LUNII

Concurs pentru elevi și studenți
organizat de Gazeta de Informatică

COMPUTER LIBRIS AGORA

MANUAL Clasa a IX -a
CONFORM PROGRAMEI ȘCOLARE

Informatică

PROFIL MATEMATICĂ-INFORMATICĂ

DOINA RANCEA
GABRIELA BĂLAN
IOANA CUCU
DIANA GRUIȚĂ

AGORA

Manualul este astfel conceput încât să permită învățarea noțiunilor prezentate cu ușurință, realizarea unor aplicații utile, individual sau în grupuri de elevi. Sunt incluse multe probleme rezolvate și propuse din care profesorul poate să aleagă, în funcție de nivelul general al clasei și importanța și dificultatea problemelor. Problemele mai dificile de la sfârșitul fiecărui capitol ar putea fi date elevilor ca "miniproiecte" de aplicații în scopul dezvoltării lor cu munca în echipă. Autorii acestui manual au căutat să ofere un instrument didactic prin utilizarea căruia să crească eficiența procesului de predare și învățare și să împartă propria lor experiență didactică, acumulată pe parcursul a peste 25 de ani.

Limbaj
Algo
fundar

Clasa a X -a

MANUAL Clasa a X -a
CONFORM PROGRAMEI ȘCOLARE

LIMBAJE DE PROGRAMARE ȘI BAZELE INFORMATICII
APLICAȚII PRACTICE DE LABORATOR

Limbajul Pascal

Algoritmi fundamentali

DOINA RANCEA

AGORA

Manualul prezintă tehnici și metode de programare ca: recursivitate, alocare dinamică, metodele backtracking, programare dinamică, greedy, divide et impera și branch and bound. Prezentarea este accesibilă, iar aplicațiile sunt diverse: grafică (tracta), jocuri, teoria grafurilor.
Manualul a fost conceput, plecând de la ediția 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a X-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Două noi apariții

Pentru comenzi și informații suplimentare ne puteți contacta la tel. 064-192422

COMPUTER LIBRIS

Vol. 10, Nr. 1/2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Cristian Cadar (București)
Ion Gelu Tomșa (Dej)
Tehnoredactare:
Mihai Uricaru, Călin Borz
Corectura:
Carmen Bucur
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Gabriela Bucșa (gbucsa@agora.ro)
Gabriel Piscupescu (gpiscup@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Annamria Pascu (apascu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Mihaela Szabo (mszabo@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

probleme.. PAG. 5

propuse pentru REZOLVARE

Olimpiada de Informatică a Europei Centrale

soluții.. PAG. 9

probleme REZOLVATE

Soluțiile problemelor propuse la Olimpiada Balcanică de Informatică 1999

concurs.. PAG. 16

Rezultatele rundei a doua

Clasament

Problemele rundei a patra

serial.. PAG. 25

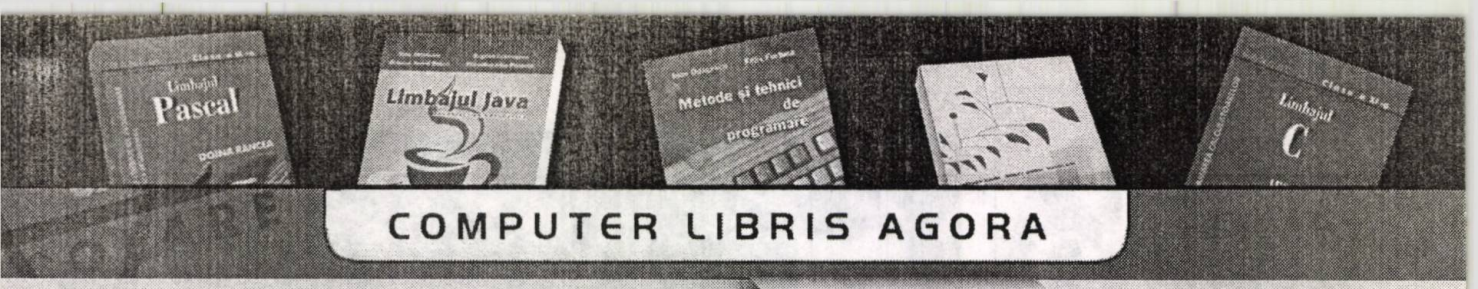
Introducere în teoria GRAFURILOR
Cristian CADAR

Pagini WEB cu JavaScript
Mihai SCORȚARU

begin.. PAG. 38

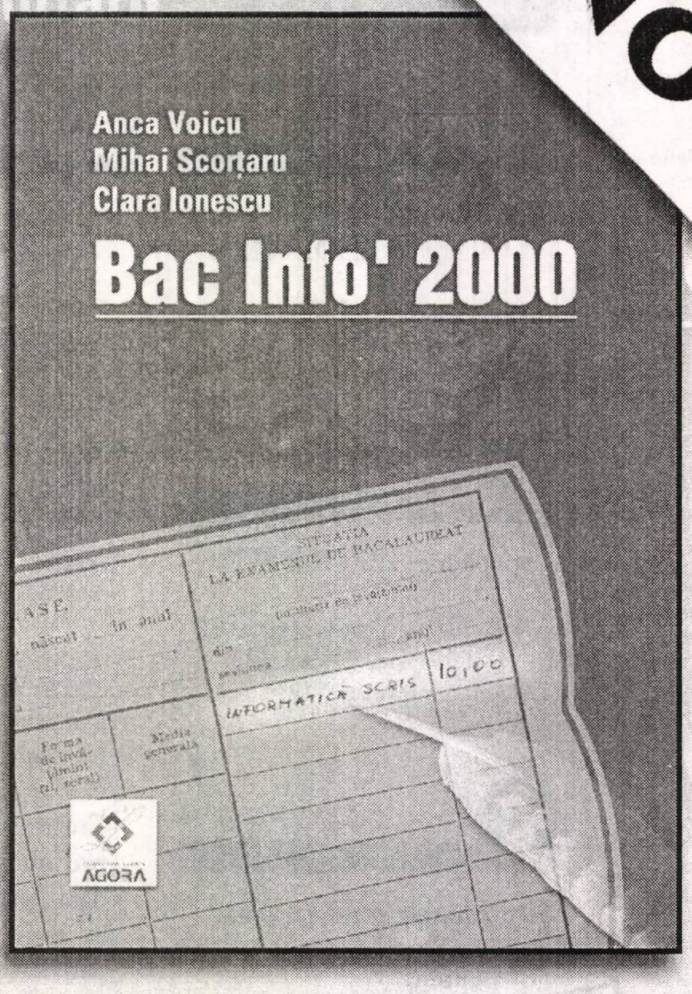
Grafica TURTLE și aplicații în PASCAL
Ion Gelu TOMȘA

Datorită disfuncționalităților apărute în încasarea banilor trimiși în contul de Trezorerie, tragerea la sorți a premiilor se amână până la data de 27.01.2000 urmând ca toți cei ce se abonează până la această dată să aibă șansa de a se număra printre câștigători.



COMPUTER LIBRIS AGORA

NOUTATE



În această cărticică veți găsi subiecte de Bac la informatică și nu doar atât.
În partea a doua, vă punem la dispoziție posibile răspunsuri, soluții ale problemelor. Deoarece limbajul de programare este la alegere, depinde de candidat pe care îl preferă. Acolo unde răspunsul depinde de limbajul ales, veți găsi soluțiile corespunzătoare prezentate și în Pascal și în C.
La sfârșitul cărții sunt patru "variante" de subiecte propuse ale căror rezolvări le veți putea citi și studia în Gazeta de Informatică.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

AGORA

Peripețiile domnului VIRUSACHE pe la concursul Bursele Agora

Este evident că tehnologia informației a schimbat semnificativ modul de viață al omenirii în ultimul deceniu. O contribuție decisivă a avut-o *Internet*-ul, acea rețea minunată cu ajutorul căreia poți obține aproape orice informație. În țările vestice deja este de neconceput ca un individ să nu aibă acces la "rețea". Toate ar fi bune și frumoase dar, ca de obicei, există și reversul medaliei. Printre multitudinea de informații disponibile pe *Internet* există deja prea multe care sunt folosite în scopuri nu tocmai ortodoxe. Există site-uri specializate care oferă informații despre modul în care poate fi "spartă" protecția unui produs software, site-uri care oferă gratuit software piratat, crack-uri pentru programe shareware care pot fi astfel "înregistrate" etc.

O categorie specială o reprezintă site-urile care oferă surse pentru viruși; acestea sunt foarte des vizitate de tinerii programatori care consideră că ***nu ești programator adevărat dacă nu ai scris cel puțin un virus!***. Urmând această lozincă, majoritatea încearcă diferite modalități de a se "exprima" (de a-și etala talentele în acest domeniu). Din păcate se pare că *GInfo* a oferit o astfel de posibilitate de exprimare de care cineva s-a hotărât să profite.

Ce s-a întâmplat? Într-o zi *minunată* ne-a venit ideea *minunată* de a lansa un concurs *minunat* prin e-mail pentru a-i ajuta pe cititori să se pregătească pentru diferite olimpiade sau alte concursuri. Începeți să vă prindeți, nu? Cineva s-a gândit că ar fi bine să trimită un program mai interesant care să rescrie boot-ul, să șteargă toate fișierele de pe disc și cine știe ce altceva. Era un virus în toată regula, scris - evident - de un începător sau un programator "adevărat" care știe programare "cam cât am uitat eu".

Sau poate că avem de-a face cu un programator generos care s-a gândit că cei de la *GInfo* nu știu cum se face un virus și s-a hotărât să ne învețe și el ceva nou. Așa se explică faptul că sursa este foarte clară, observându-se încă de la început, intențiile programului. Dacă aceasta este situația, nu putem decât să-i mulțumim respectivului (respectivei) pentru bunele intenții, dar să-i spunem că nu era cazul, deoarece asemenea programe putem realiza și noi și, cu toată modestia care ne caracterizează, efectele acestora ar fi fost mult mai spectaculoase.

Totuși am fost și noi curioși să vedem ce știe face *Dl. Virusache* (așa l-am numit noi în redacție). Într-adevăr programul reușea să rescrie boot-ul fără ca antiviruții care rulau în background să detecteze această operație. O realizare spectaculoasă, nu-i așa? Totuși, dacă lansăm în executare un anumit antivirus acesta detecta imediat că boot-ul a fost modificat și îl putea înlocui cu un back-up. Așa că, marea realizare se dovedește a fi inutilă. Apoi am hotărât să lăsăm virusul să infecteze un calculator, dar se pare că acesta nu l-a prea ascultat pe "programator". Sau poate că virusul infecta atât de subtil fișierele executabile încât la o comparare cu cópii ale acestora salvate anterior, cele două versiuni erau absolut identice. Sau poate ne-am înșelat și de această dată și virusul a modificat de fapt și programul care compara fișierele...

Îmi pare rău pentru cel (cea) care a semnat cu numele Rodica-Cristina Vasilescu, dar de această dată, tentativa a fost un eșec total. Nu pot decât să-i urez *Succes în continuare!*; s-ar putea ca, în alte situații, să reușească să creeze cuiva probleme trimițând astfel de cadouri. Eu nu pot decât să-i mulțumesc pentru că ne-a oferit un prilej de distracție pe cinste în redacție și îi promit că *Dl. Virusache* va rămâne pentru multă vreme în memoria noastră, deoarece, la fel ca și primul sărut, primul virus primit (mai ales în locul rezolvării unei probleme nevinovate) nu se uită niciodată.

S-ar putea ca scopul respectivului (respectivei) să fie totuși atins pentru că o țară întreagă a aflat de isprava lui (ei). Să fi fost râvnită chiar această publicitate? Tot ce se poate...

Probabil că acest editorial îi va determina pe mulți să-și încerce și ei norocul cu asemenea "glume". Nu aveți decât... Vă promitem că celui dintâi "concurent" care va reuși să ne determine să rulăm un astfel de program (fără să ne dăm seama că este de fapt un virus) îi va fi dedicat un întreg număr al *Gazetei de Informatică*.

În final nu pot decât să urez succes tuturor participanților la consursul *Bursele Agora* și îmi cer scuze pentru faptul că, din motive lesne de înțeles, a trebuit să interzicem folosirea secvențelor în limbaj de asamblare în cadrul soluțiilor la problemele de concurs.

Mihai Scorțaru

15 Ianuarie 1900

(Se pare că dl. Virusache e aproape la fel de periculos ca și Y2K)

COMPUTER LIBRIS AGORA

NOUTATE

Culegerea constituie un instrumentar pentru cei care doresc să intre în posesia secretelor proiectării algoritmilor. Va fi util când vă pregătiți pentru teză, pentru examenul de bacalaureat sau pentru admitere. Veți găsi în culegere și probleme care pot fi analizate și rezolvate în cercurile de elevi și care vor fi utile în pregătirea pentru concursurile școlare.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

PROBLEME propuse pentru REZOLVARE

Olimpiada de Informatică a Europei Centrale a fost organizată anul acesta în orașul Brno, în Cehia. Vă propunem să încercați să rezolvați problemele cu care s-au confruntat concurenții la această competiție.

P010001: Livrări

Șeful unui magazin sortează toate tipurile de produse în ordine alfabetică, pe baza etichetelor lor.

Toate produsele ale căror etichete încep cu aceeași literă sunt așezate în același depozit ce va fi etichetat cu această literă.

Pe parcursul unei zile, șeful primește și înregistrează comenzi de produse care vor fi livrate din depozite. Fiecare cerere de livrare se referă la un singur tip de produse. Șeful efectuează livrările în ordinea înregistrării lor.

Se cunosc cererile de livrare care urmează a fi satisfăcute într-o zi, dar nu se cunoaște ordinea în care au fost înregistrate.

Determinați toate modalitățile posibile de vizitare a depozitelor în scopul de a aduna toate produsele solicitate unul câte unul.

Date de intrare

Fișierul de intrare `ORDERS.IN` conține o singură linie pe care sunt scrise toate etichetele produselor solicitate (ordinea etichetelor este oarecare).

Fiecare produs este reprezentat prin prima literă a etichetei. Sunt folosite doar literele mici ale alfabetului latin.

Numărul livrărilor nu va fi mai mare decât 200.

Date de ieșire

Fișierul de ieșire `ORDERS.OUT` va conține toate modalitățile posibile de vizitare a depozitelor:

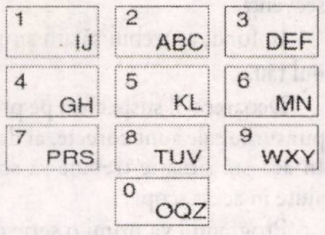
- Fiecare depozit este reprezentat printr-o literă a alfabetului latin - prima literă a etichetelor din acel depozit
- Fiecare succesiune de vizite se va scrie în fișierul de ieșire o singură dată, pe câte o linie.
- Acestea vor fi scrise în fișier în ordine alfabetică (vezi exemplul).
- Nu există fișiere de ieșire cu dimensiuni mai mari decât 2MB.

Exemplu

ORDERS.IN	ORDERS.OUT
bbjd	bbdj
	bbjd
	bdbj
	bdjb
	bjbd
	bfdb
	dbbj
	dbjb
	djbb
	jbbd
	jbdb
	jdbb

P010002: Numere de telefon

Deoarece numerele de telefon sunt formate frecvent din foarte multe cifre, în scopul de a le memora mai ușor, se asociază litere la cifre, ca în figură. În acest fel fiecare cuvânt sau grup de cuvinte poate fi asociat unui singur număr de telefon. De exemplu, numărului de telefon `941837296` i se asociază cuvântul `WHITEPAWN`, iar numărului de telefon `2855304` i se va asocia cuvântul `BULLDOG`.



Scrieți un program care determină o secvență formată dintr-un număr minim de cuvinte care corespunde numărului de telefon dat și este formată folosind cuvinte dintr-o listă dată. Corespondența dintre cifre și litere se găsește în figură.

Date de intrare

Prima linie a fișierului `PHONE.IN` conține numărul de telefon a cărui codificare se caută. Numărul poate fi format din cel mult `100` de cifre. Cea de-a doua linie conține numărul

cuvintelor din dicționar (maxim 50000 de cuvinte). Restul liniilor conțin câte un cuvânt format din cel mult 50 de litere mici ale alfabetului latin. Dimensiunea maximă a fișierului de intrare nu depășește 300KB.

Date de ieșire

Singura linie a fișierului de ieșire **PHONE.OUT** va conține secvența formată din numărul minim de cuvinte determinată de program.

- Cuvintele vor fi separate printr-un singur spațiu.
- Dacă pentru un anumit conținut al fișierului de intrare problema nu are soluție, în fișierul de ieșire se va scrie `'No solution.'`.
- Dacă pentru un anumit conținut al fișierului de intrare, problema are mai multe soluții, având același număr minim de cuvinte, în fișierul de ieșire se va scrie una singură.

Exemple

PHONE.IN	PHONE.OUT
7325189087 5 it your reality real our	reality our

Secvența `real it your` are un număr mai mare de cuvinte.

Pentru numărul 4294967296 singurul conținut corect al fișierului de ieșire este: `No solution` deoarece nici un cuvânt dat nu conține litera `g`, respectiv `h`, necesare codificării cifrei 4.

P010003: Joc de paritate

Tu și prietenul tău jucați următorul joc. Prietenul tău se gândește la un șir de cifre 0 și 1 pe care îl reține. Tu poți alege o secvență de poziții consecutive (de exemplu, începând cu cea de-a treia cifră până la a cincea inclusiv) și îl poți întreba dacă această parte din șirul ales conține un număr par sau un număr impar de cifre 1. Pritenul îți răspunde la întrebare și tu poți pune altă întrebare pentru o altă secvență.

În fond, ar trebui să afli șirul la care s-a gândit prietenul tău.

Deoarece îl suspectezi pe prietenul tău că nu toate răspunsurile sale sunt corecte, ai dori să-i dovedești că minte. În această situație trebuie să scrii un program care să te ajute în acest scop.

Programul va primi o serie de întrebări de ale tale împreună cu răspunsurile primite. Sarcina programului este de a afla primul răspuns care rezultă a fi greșit, adică există un șir de cifre de 1 și 0 pentru care toate răspunsurile precedente sunt adevărate, dar răspunsul curent este fals, altfel spus, nu există nici o secvență pentru care să fie corect și acest răspuns.

Date de intrare

Prima linie a fișierului **PARITY.IN** conține un singur număr, reprezentând lungimea șirului de 0 și 1. Această lungime este mai mică sau egală cu 1000000000. Pe cea de-a doua linie se află un număr întreg pozitiv care reprezintă numărul de întrebări și răspunsuri. Numărul întrebărilor ce au alăturate răspunsurile este mai mic sau egal cu 5000. Restul liniilor din fișier conțin întrebările și răspunsurile. Fiecare linie conține o singură întrebare și răspunsul aferent: două numere naturale (prima și ultima poziție a secvenței din șirul de cifre ales) și un cuvânt care este fie `'even'`, fie `'odd'` (cuvânt care se referă la paritatea numărului de cifre de 1 în secvența aleasă, par, respectiv impar).

Date de ieșire

Fișierul de ieșire **PARITY.OUT** conține o singură linie pe care se va scrie un singur număr întreg X. Acest număr semnifică faptul că există un șir de 1 și 0 care satisfac primele X condiții de paritate - răspunsuri -, dar nu satisfac primele X+1 condiții.

Dacă toate răspunsurile se dovedesc a fi corecte, adică există cel puțin un șir de 1 și 0 care satisface toate condițiile date, numărul X va fi egal cu numărul de întrebări din fișier.

Exemple

PARITY.IN	PARITY.OUT
10 5 1 2 even 3 4 odd 5 6 even 1 6 even 7 10 odd	3

PARITY.IN	PARITY.OUT
10 5 1 2 even 1 4 even 2 4 odd 1 10 even 3 10 even	5

P010004: Vizita orașului

O agenție de voiaj dorește să ofere solicitanților un traseu de vizitare a orașului. Deoarece sunt și alte programe distractive de realizat, agenția a luat o decizie naspa și anume se caută cel mai scurt traseu care începe și se termină în același loc. Programul vostru trebuie să determine un astfel de traseu.

În oraș există N noduri numerotate de la 1 la N și M muchii numerotate de la 1 la M. Două noduri pot fi legate prin una sau mai multe muchii; aceste muchii corespund străzilor din oraș și toate sunt cu circulație în ambele sensuri. Nu există muchie de la un nod la el însuși. Fiecare traseu

este format dintr-o secvență de muchii $y_1, ..., y_k$, $k>2$. Muchia y_i $(1 \le i \le k-1)$ leagă nodurile x_i și x_{i+1}, muchia y_k leagă nodurile x_i și x_{i+1}. Toate nodurile de pe traseu trebuie să fie distincte. Lungimea traseului este suma tuturor muchiilor de pe traseu: $L(y_1)+L(y_2)+...+L(y_k)$ unde $L(y_i)$ este lungimea muchiei y_i $(1 \le i \le k)$.

Programul vostru trebuie să determine un traseu de lungime minimă care pleacă dintr-un nod, se întoarce în acest nod și nu trece neapărat prin toate nodurile. În cazul în care nu este posibilă determinarea, se va afișa mesajul `'No solution'`.

Date de intrare

- Prima linie a fișierului de intrare **TRIP.IN** conține două numere pozitive întregi: numărul nodurilor $N \le 100$ și numărul muchiilor $M \le 10000$.
- Pe fiecare dintre următoarele linii este specificată câte o muchie prin trei numere pozitive: cele două noduri de la capetele muchiei și lungimea asociată acesteia (număr pozitiv întreg mai mic decât 500).

Date de ieșire

- Fișierul de ieșire **TRIP.OUT** conține o singură linie.
- Dacă problema nu are soluție, conținutul va fi mesajul `'No solution.'`.
- Dacă există traseu care respectă cerințele problemei, în fișier se va scrie secvența nodurilor de pe traseu în ordinea de parcurgere a traseului (numerele $x_1, ..., x_k$ din enunț), separate printr-un singur spațiu.
- Dacă există mai multe soluții, în fișier se va afișa una singură.

Exemple

TRIP.IN

```
5 7
1 4 1
1 3 300
3 1 10
1 2 16
2 3 100
2 5 15
5 3 20
```

TRIP.OUT (o soluție posibilă)

```
1 3 5 2
```

TRIP.IN

```
4 3
1 2 10
1 3 20
1 4 30
```

TRIP.OUT (singura soluție corectă)

```
No solution.
```

P010005 Pe tabla de șah

Pe o tablă de șah, având dimensiunea 4×4 există 8 piese albe și 8 piese negre, adică o piesă pe fiecare pătrățel. O asemenea configurație de piese se numește o stare a jocului. Două piese sunt vecine dacă sunt plasate pe pătrățele având o latură comună (adică sunt vecine pe orizontală sau pe verticală, dar nu pe diagonală). Rezultă că fiecare piesă poate avea cel mult patru vecini. Singura mutare permisă este interschimbarea a două piese vecine. Programul trebuie să determine succesiunea cea mai scurtă de mutări care transformă starea inițială a jocului în starea finală. Ambele stări se dau.

Date de intrare

Starea inițială a jocului este descrisă în primele linii ale fișierului **GAME.IN**. Pe fiecare linie sunt scrise 4 caractere, care precizează culoarea fiecărei piese aferente liniilor de pe tablă de sus în jos, respectiv de la stânga la dreapta. Caracterul `'0'` reprezintă o piesă albă, iar caracterul `'1'` corespunde unei piese negre. Între caracterele care descriu piesele de pe o linie a tablei nu există separatori. A cincea linie este vidă. Pe următoarele 4 linii se descrie starea finală a jocului în același mod.

Date de ieșire

Prima linie a fișierului de ieșire **GAME.OUT** va conține numărul minim de mutări.

Următoarele linii trebuie să conțină descrierea mutărilor în ordinea efectuării lor. O linie descrie o singură mutare și conține patru numere întregi pozitive, separate printr-un singur spațiu în următorul format: `R1 C1 R2 C2`.

Acestea reprezintă coordonatele pieselor vecine, interschimbate pe parcursul jocului. `[R1 C1]` respectiv `[R2 C2]` reprezintă pătrățele pe care sunt situate piesele interschimbate. `R1` sau `R2` reprezintă numărul de ordine al liniei, iar `C1`, respectiv `C2` reprezintă numărul de ordine al coloanei. Liniile tablei sunt numerotate începând cu 1 (cea mai din stânga coloană) până la patru (cea mai din dreapta). Coordonatele colțului stânga sus sunt `[1 1]`.

Dacă există mai multe soluții cu același număr minim de mutări, se va afișa una singură.

Exemplu

GAME.IN

```
1111
0000
1110
0010

1010
0101
1010
0101
```

GAME.OUT (o soluție posibilă)

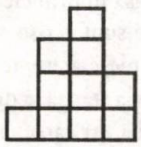
```
4
1 2 2 2
1 4 2 4
3 2 4 2
4 3 4 4
```

P000106 Jocul de-a constructorul

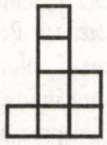
Johny dorește să construiască un orășel mare din cuburile sale. La baza orășelului are o tablă rectangulară care are o lățime pe care încap K cuburi și o lungime pe care se pot așeza L cuburi. Înainte de a se apuca de construit, Johny desenează planul orașului și împarte tabla în $K \times L$ pătrățele. El dorește să așeze turnuri formate din unul sau mai multe cuburi pe câteva dintre pătrățele. Unele pătrățele pot rămâne

libere. Deoarece tabla e mare, Johny nu poate planifica în mod exact numărul de cuburi care vor fi așezate pe câte un pătrățel. El a stabilit imaginea din față și cea lateral-dreapta a construcției și a desenat aceste două perspective (proiecții bidimensionale).

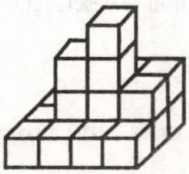
Tatăl lui Johny se teme că nu are destui bani pentru a cumpăra destule cuburi și vă roagă să scrieți un program care determină numărul minim și cel maxim de cuburi



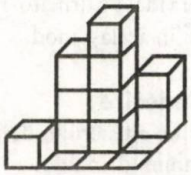
Vedere din față



Vedere din lateral dreapta



Orașul maximal



Orașul minimal

necesare realizării planului pustiului. Deoarece Johny poate greși desenele planului, programul va trebui să stabilească dacă orașul descris este realizabil sau nu.

Date de intrare:

Prima linie a fișierului `TOWN.IN` conține două numere întregi pozitive `K`, `L` - lățimea respectiv lungimea tablei (exprimate în număr de cuburi). Nici una dintre dimensiuni nu este mai mare decât `100000`.

Următoarele linii ale fișierului de intrare conțin descrierea imaginii din față a orășelului. Descrierea este realizată prin șirul înălțimilor clădirilor vizibile pentru fiecare pătrățel de la stânga la dreapta (înălțimea se precizează de asemenea prin numărul de cuburi). Pe fiecare linie există un singur număr. Primele K linii descriu perspectiva din față, iar restul de L linii descriu perspectiva din lateral-dreapta.

În această descriere înălțimile sunt precizate de la linia din față spre cea din spate. Se știe că nu există clădire cu înălțimea mai mare decât 5000 de cuburi. Numărul maxim de cuburi necesare construirii orașului nu va depăși 2000000000.

Date de ieșire

- Fișierul de ieșire `TOWN.OUT` conține o singură linie.
- Dacă orașul nu poate fi construit pe baza desenelor lui Johny, în fișier se va scrie mesajul `'No solution.'`.
- Altfel, în fișier se vor scrie două numere, pe o singură linie, separate printr-un singur spațiu. Primul număr reprezintă numărul minim de cuburi necesare, iar al doilea reprezintă numărul maxim.

Exemple

TOWN.IN	TOWN.OUT
4 3 1 3 4 2 1 4 2	10 21

TOWN.IN	TOWN.OUT
2 2 4 1 1 3	No solution.

News...

Intel Pentium III la 800 MHz

Firma Intel a lansat pe data de 15 decembrie 1999 două noi versiuni ale procesoarelor Pentium III având viteze de 750 MHz, respectiv 800 MHz. Deși lansarea Pentium III 750 era inițial programată pentru 10 ianuarie 2000, iar lansarea Pentium III 800 pentru sfârșitul primului trimestru al anului 2000, cei de la Intel au decis că trebuie să intre în anul 2000 ca lideri în "războiul procesoarelor", motiv pentru care au lansat mai devreme noile procesoare.

AMD Athlon la 800 MHz

La puțin timp după ce Intel anunță victorioasă procesorul Pentium III la 800 MHz, AMD îi răspunde cu un Athlon echivalent. Așa cum se aștepta mai toată lumea, compania AMD a lansat noul procesor Athlon, care funcționează la o frecvență de 800 MHz, la Consumer Electronics Show (CES) din Las Vegas. Noul Athlon, la fel ca și predecesorul său de 750 MHz are la bază tehnologia de 0,18 microni și este produs de unitatea din Austin, Texas a companiei AMD.

SOLUȚIILE problemelor propuse

Vă prezentăm în continuare soluțiile problemelor de la Olimpiada Balcanică de Informatică 1999. Rezolvările aparțin lui Tudor Ioan Leu din Focșani, medaliat cu argint la BOI '99

P089901: Drumuri pavate

Cele două condiții impuse mulțimii de drumuri pavate descriu un arbore parțial. Așadar, se cere numărul total de arbori parțiali ai grafului dat la intrare.

Pentru generarea arborilor am folosit metoda backtracking. Principala dificultate constă în evitarea obținerii aceluiași arbore de mai multe ori. Pentru a elimina această posibilitate, sunt suficiente condițiile:

1) Rădăcina oricărui arbore este vârful numerotat cu 1.
2) Fiii unui nod deja existent în arbore sunt adăugați respectând o relație de ordine. În implementarea prezentată mai jos s-a folosit ordinea în care sunt dați la intrare vecinii nodului respectiv. Cu alte cuvinte, dacă nodul k face parte din arborele curent și în arbore am adăugat ca fii ai lui k vârfuri care se află în primele vk poziții în lista lui de vecini, nu-i vom putea adăuga fii decât dintre vecinii aflați pe o poziție mai mare decât vk.

Listing: DRUMURI.PAS

```
const ni='INPUT.TXT';
      no='OUTPUT.TXT';
type sir=array[1..20] of Byte;
var v:array[1..20] of sir;
    nv,x,g:sir;
    f:array[1..20] of Boolean;
    i,j,k,n,r:Byte;
    sol:Longint;

procedure citeste;
begin
  Assign(input,ni); Reset(input);
  Readln(n,r);
  for i:=1 to n do nv[i]:=0;
    for k:=1 to r do
    begin
      Readln(i,j); Inc(i); Inc(j);
      Inc(nv[i]); v[i,nv[i]]:=j;
      Inc(nv[j]); v[j,nv[j]]:=i
    end
end;

procedure bkt(k,tk,vk:Byte);
begin
  if k=n+1 then Inc(sol)
  else
    while tk<k do
      if vk=nv[x[tk]]+1
      then begin Inc(tk); vk:=1 end
      else
      begin
        if not f[v[x[tk],vk]]
        then
        begin
          x[k]:=v[x[tk],vk];
          f[x[k]]:=true;
          bkt(k+1,tk,vk+1);
          f[x[k]]:=false
        end;
        Inc(vk)
      end
end;

procedure rezolva;
begin
  for i:=2 to n do
  begin g[i]:=nv[i]; f[i]:=false end;
  g[1]:=nv[1];
  f[1]:=true;
  sol:=0;
  x[1]:=1;
  bkt(2,1,1)
end;
```

```
procedure scrie;
begin
  Assign(output,no); Rewrite(output);
  Writeln(sol);
  Close(output)
end;

Begin
  citeste;
  rezolva;
  scrie
End.
```

P089902: Jocul flip

Algoritmul de rezolvare se bazează pe faptul că numărul de coloane al tablei este mic și, în consecință, se pot genera toate variantele de întoarcere a coloanelor.

Variantele de întoarcere a coloanelor sunt reprezentate pe ultimii 9 biți ai variabilei contor k. Pentru fiecare dintre aceste variante, se parcurge toată tabla de joc și se calculează pentru fiecare linie numărul de piese care au la vedere fața neagră, apoi se decide dacă linia este întoarsă sau nu, în funcție de acest număr: mai mare decât 4 sau, respectiv, mai mic sau egal cu 4.

Pentru o tablă cu m linii și n coloane, complexitatea algoritmului este $O(2 \cdot m \cdot n \cdot min\{m,n\})$.

Listing: FLIP.PAS

```
const ni='INPUT.TXT';
      no='OUTPUT.TXT';
      q:array['0'..'1'] of Byte=(0,1);
var a:array[1..1000,1..9] of Byte;
    fc:array[1..9] of Byte;
    i,j,k,l,m,s,min:Word;

procedure citeste;
var c,d:Char;
begin
  Assign(input,ni); Reset(input);
  Readln(m);
  for i:=1 to m do
  begin
    for j:=1 to 9 do
    begin
      Read(c,d);
      a[i,j]:=q[c];
    end;
    Read(d)
  end
end;

procedure rezolva;
begin
  min:=9*m+1;
  for k:=0 to 511 do
  begin
    for j:=1 to 9 do
      fc[j]:=k shr (j-1) and 1;
    s:=0;
    for i:=1 to m do
    begin
      l:=0;
      for j:=1 to 9 do
        l:=l+(a[i,j] xor fc[j]);
      if l<5 then s:=s+l
             else s:=s+9-l
    end;
    if s<min then
    begin Writeln(k); min:=s end
  end
end;

procedure scrie;
begin
  Assign(output,no); Rewrite(output);
  Writeln(min);
  Close(output)
end;

Begin
  citeste;
  rezolva;
  scrie
End.
```

P089903: De pază în galeria de arte

Pentru fiecare perete al galeriei se poate determina regiunea din care acesta este vizibil. Intersecția acestor regiuni va fi formată din toate punctele în care paznicul ar putea fi așezat pentru a supraveghea toată galeria.

Fie $P_i(x_i, y_i)$ și $P_{i-1}(x_{i-1}, y_{i-1})$ capetele unui segment din conturul galeriei.

1) Dacă $x_i=x_{i-1}$ și $y_i<y_{i-1}$, atunci segmentul este vizibil din semiplanul închis constituit din dreapta suport a segmentului și din punctele din dreapta acesteia.
2) Dacă $x_i=x_{i-1}$ și $y_i>y_{i-1}$, atunci segmentul este vizibil din semiplanul închis constituit din dreapta suport a segmentului și din punctele din stânga acesteia.
3) Dacă $y_i=y_{i-1}$ și $x_i<x_{i-1}$, atunci segmentul este vizibil din semiplanul închis constituit din dreapta suport a segmentului și din punctele de sub aceasta.
4) Dacă $y_i=y_{i-1}$ și $x_i>x_{i-1}$, atunci segmentul este vizibil din semiplanul închis constituit din dreapta suport a segmentului și din punctele de deasupra acesteia.

Aceste observații sunt valabile doar în cazul în care colțurile galeriei sunt date în sensul acelor de ceasornic.

Dacă intersecția semiplanelor determinate este vidă, nu există soluție.

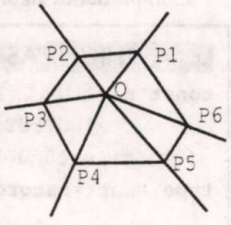
Altfel, intersecția poate fi un punct, un segment sau un interior de dreptunghi.

Listing: GALERIE.PAS

```
const ni='INPUT.TXT';
      no='OUTPUT.TXT';
type colt=record x,y:Integer end;
var g:array[1..1001] of colt;
    n,i,maxx,minx,maxy,miny:Integer;

function min(a,b:Integer):Integer;
begin
  if a<b then min:=a else min:=b;
end;

function max(a,b:Integer):Integer;
begin
  if a>b then max:=a
         else max:=b;
end;

Begin
  Assign(input,ni); Reset(input);
  Readln(n);
  for i:=1 to n do
    Readln(g[i].x,g[i].y);
  g[n+1]:=g[1];
  minx:=-1; miny:=-1;
  maxx:=1001; maxy:=1001;
  for i:=1 to n do
    if g[i].x=g[i+1].x
    then
      if g[i].y<g[i+1].y
      then minx:=max(minx,g[i].x)
      else maxx:=min(maxx,g[i].x)
    else
    if g[i].x<g[i+1].x
      then maxy:=min(maxy,g[i].y)
      else miny:=max(miny,g[i].y);
    Assign(output,no); Rewrite(output);
    if (minx>maxx) or (miny>maxy)
    then Writeln('NO POINT')
    else Writeln((minx+maxx) div 2,
                 ' ',(miny+maxy) div 2)
End.
```

P089904: Statistici la poliție

Ideea de rezolvare este următoarea: se alege un punct O în interiorul poligonului; se observă că semidreptele determinate de punctul O și vârfurile poligonului împart planul în n regiuni disjuncte.



Problema revine deci la:

a) determinarea regiunii căreia îi aparține punctul de test A și
b) verificarea faptului dacă A se află sau nu de aceeași parte a laturii corespunzătoare regiunii cu punctul O.

Dacă punctul A se află de aceeași parte a laturii cu O, sau dacă este chiar pe latură, el corespunde cerinței inițiale. Altfel, el se află în afara poligonului.

Mai rămâne de rezolvat cazul a). Se observă că măsurile unghiurilor pe care le formează semidreptele (OP_i cu axa absciselor formează un șir de valori crescătoare. Punctul a) al rezolvării este echivalent cu găsirea intervalului căruia îi aparține măsura unghiului format de semidreapta (OA cu abscisa. Folosind căutarea binară, acest interval poate fi găsit într-un timp $O(\log N)$.

Așadar, complexitatea algoritmului este $O(M \log N)$.

Listing: POLITIE.PAS

```
const ni='INPUT.TXT';
      no='OUTPUT.TXT';
      mare=10000;
type punct=record x,y:Longint end;
     camp=record u:Real; v:Integer end;
     lcamp=array[1..1000] of camp;
var g:array[1..1000] of punct;
    gu:array[0..1000] of Real;
    c:lcamp; a:punct;
    i,j,k,m,n,crime,test:Integer;
    cgx,cgy,u,t:Real;

procedure citeste;
begin
  Assign(input,ni); Reset(input);
  Readln(n);
  for i:=1 to n do Readln(g[i].x,g[i].y);
  Readln(m)
end;

function det(a,b,c:punct):Longint;
begin
  det:=a.x*b.y+b.x*c.y+c.x*a.y-a.y*b.x-
                              b.y*c.x-c.y*a.x
end;

procedure init;
begin
  cgx:=0; cgy:=0;
  for i:=1 to n do
  begin
    cgx:=cgx+g[i].x; cgy:=cgy+g[i].y;
  end;
  cgx:=cgx/n; cgy:=cgy/n; k:=0; gu[0]:=pi;
  for i:=1 to n do
    if Abs(cgx-g[i].x)<0.001 then
      if cgy<g[i].y then gu[i]:=Pi/2
                    else gu[i]:=3*Pi/2
    else
    begin
      t:=(g[i].y-cgy)/(g[i].x-cgx);
      gu[i]:=Arctan(t);
```

```
      if cgx>g[i].x
      then gu[i]:=gu[i]+pi
      else
        if gu[i]<0
        then gu[i]:=gu[i]+2*pi;
          if gu[i]<gu[k] then k:=i
    end;
  if k=n then i:=1 else i:=k+1;
  if k=1 then j:=n else j:=k-1;
  if gu[i]<gu[j] then
  begin
    i:=0;
    while i<n do
    begin
      Inc(i); c[i].u:=gu[k]; c[i].v:=k;
      if k=n then k:=1
      else Inc(k)
    end;
  end
  else begin
         i:=0;
         while i<n do
         begin
           Inc(i);
           c[i].u:=gu[k]; c[i].v:=k;
           if k=1 then k:=n
                  else Dec(k)
         end
       end
end;

procedure rezolva;
begin
  init; crime:=0;
  Assign(output,no); Rewrite(output);
  for test:=1 to m do
  begin
    Dec(m); Readln(a.x,a.y);
    if Abs(cgx-a.x)<0.001 then
      if a.y>cgy then u:=Pi/2
                 else u:=3*Pi/2
    else
    begin
      t:=(a.y-cgy)/(a.x-cgx); u:=Arctan(t);
      if a.x<cgx then u:=u+pi
      else if u<0 then u:=u+2*pi;
    end;
    if (u<=c[1].u) or (u>=c[n].u) then
    begin
      if det(a,g[c[n].v],g[c[1].v])>=0 then
        Inc(crime)
    end
    else
    begin
      i:=1; j:=n;
      while i<j-1 do
        if u>=c[(i+j) div 2].u then
          i:=(i+j) div 2
        else j:=(i+j) div 2-1;
      j:=c[i+1].v; i:=c[i].v;
      if det(a,g[i],g[j])>=0
        then Inc(crime)
    end
  end
end;

procedure scrie;
begin Writeln(crime) end;

Begin
  citeste;
  rezolva;
  scrie
End.
```

P089905: Râul Highwater

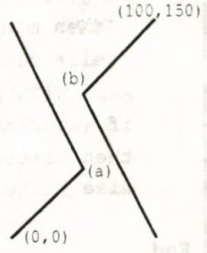
Metoda de rezolvare folosită este *Divide et Impera*.

Se verifică dacă traseul direct între punctul de plecare și cel de sosire este împiedicat de vreun colț al unui mal. Dacă nu, atunci drumul este, evident, segmentul determinat de cele două puncte. Altfel, se selectează un colț O (despre care se știe cu siguranță că aparține traseului final) și se determină drumul în două etape: de la punctul de plecare la O și de la O la punctul de sosire.

Tipurile de obstacole care pot apărea sunt:

(a) colțuri ale malului vestic care se află în dreapta traseului direct;

(b) colțuri ale malului estic care se află în stânga traseului direct.



Selectăm "cel mai din dreapta" obstacol de tip (a) și "cel mai din stânga" obstacol de tip (b). (Atenție! Aceste exprimări nu se referă la abscisele punctelor de pe maluri, ci la unghiurile pe care le formează cu abscisa semidreptele determinate de punctul de plecare și aceste puncte). Obstacolul O căutat este cel cu ordonata mai mică dintre cele două selectate.

Complexitatea algoritmului descris este $O((M+N)^2)$.

Listing: HIGH.PAS

```
const ni='INPUT.TXT';
      no='OUTPUT.TXT';
      mare=1000000.0;
type punct=record
            x,y:Longint;
            mal:Byte
          end;
```

```
     list=array[1..4000] of Integer;
     listp=array[1..4000] of punct;
var p:listp; q:^listp;
    t:list;
    i,j,k,l,m,n,e,f:Longint;
    tg,u,ctg,st,dr,ust,udr:Real;

procedure citeste;
begin
  Assign(input,ni); Reset(input);
  Readln(m,n);
  for i:=1 to m do
    with p[i] do
    begin
      Readln(x,y); mal:=1
    end;
  for i:=m+1 to m+n do
  with p[i] do
  begin
    Readln(x,y); mal:=2
  end
end;

procedure QuickSort(var A:List;
                    Lo,Hi:Integer);

  procedure Sort(l,r:Integer);
  var i,j,x,y:Integer;
  begin
    i:=l; j:=r; x:=a[(l+r) div 2];
    repeat
      while p[a[i]].y<p[x].y do Inc(i);
      while p[x].y<p[a[j]].y do Dec(j);
      if i<=j
      then
      begin
        y:=a[i]; a[i]:=a[j]; a[j]:=y;
        i:=i+1; j:=j-1
      end
    until i>j;
    if l<j then Sort(l,j);
    if i<r then Sort(i,r)
  end;

begin
  Sort(Lo,Hi)
end;

procedure gaseste(k,l:Integer);
var e,f:Integer;
begin
  if k=l-1 then
    begin t[k]:=l; Exit end;
  if p[k].x=p[l].x
  then u:=Pi/2
  else
  begin
    tg:=(p[l].y-p[k].y)/(p[l].x-p[k].x);
    u:=Arctan(tg);
    if u<0 then u:=u+Pi
  end;
  st:=-mare; e:=0;
  dr:=mare; f:=0;
  for i:=k+1 to l-1 do
  if p[k].y<>p[i].y
  then
  begin
    ctg:=(p[i].x-p[k].x)/(p[i].y-p[k].y);
    if p[i].mal=1 then
    begin
      if ctg>=st then
      begin
        st:=ctg; e:=i
      end;
    end
    else
      if ctg<=dr then
      begin
        dr:=ctg; f:=i
      end
  end;
  if e=0 then ust:=Pi
  else
    if st=0 then ust:=Pi/2
    else
    begin
      ust:=Arctan(1/st);
      if ust<0 then ust:=ust+Pi
    end;
  if f=0 then udr:=0
  else
    if dr=0 then udr:=Pi/2
    else
    begin
      udr:=Arctan(1/dr);
      if udr<0 then udr:=udr+Pi;
    end;
  if (ust>=u) and (udr<=u) then t[k]:=l
  else
    if ust>=u then
    begin gaseste(k,f); gaseste(f,l) end
    else
      if udr<=u then
      begin gaseste(k,e); gaseste(e,l) end
      else
      if e<f then
      begin gaseste(k,e); gaseste(e,l) end
      else
      begin gaseste(k,f); gaseste(f,l) end
end;
```

```
procedure rezolva;
begin
  for i:=1 to m+n do t[i]:=i;
  quicksort(t,1,m+n);
  if t[2]=1 then
  begin t[2]:=t[1]; t[1]:=1 end;
  if t[m+n]<>m+n-1 then
  begin t[m+n-1]:=t[m+n]; t[m+n]:=m+n end;
  New(q);
  for i:=1 to m+n do q^[i]:=p[t[i]];
  p:=q^; Dispose(q);
  gaseste(1,m+n);
  t[m+n]:=0;
end;

procedure scrie;
begin
  Assign(output,no); Rewrite(output);
  i:=1;
  while i>0 do
  begin
    Writeln(p[i].x,' ',p[i].y); i:=t[i]
  end
end;

Begin
  citeste;
  rezolva;
  scrie
End.
```

P089906: Cunoștințe

Este evident faptul că poate fi așezată prima în coadă orice persoană ai cărei cunoscuți se cunosc între ei. Ar putea să apară situația în care există mai multe persoane care respectă această condiție. În acest caz putem alege oricare dintre aceste persoane deoarece restul persoanelor pot fi așezate în imediata apropiere, deoarece persoanele rămase vor putea fi așezate în coadă ca și cum acea persoană ar fi prima (eliminând astfel persoana deja așezată anterior în coadă). De aceea după ce am așezat o persoană în coadă problema se reduce la cea inițială pentru restul persoanelor. Așadar, la fiecare pas, vom încerca să așezăm pe o poziție o persoană ai cărei cunoscuți care nu se află deja în coadă se cunosc între ei. Pentru simplificare, în momentul în care vom așeza o persoană în coadă, o vom șterge din lista cunoscuților tuturor celorlalte persoane.

Listing: CUNOSTINTE.CPP

```
#include <stdio.h>
#include <stdlib.h>

int m,n,a[51][51],b[50][50],nr[50],sol[50],
                                    ut[50];

void lecture(void){
  int x,y; FILE *f=fopen("INPUT.TXT","rt");
  fscanf(f,"%d%d",&n,&m);
  for (int i=0;i<m;i++){
    fscanf(f,"%d%d",&x,&y);
    a[x-1][y-1]=a[y-1][x-1]=1;
    b[x-1][nr[x-1]++]=y-1;
    b[y-1][nr[y-1]++]=x-1;
  }
  for (i=0;i<n;i++) sol[i]=-1;
}

int possible(int k){
  if (ut[k]) return 0;
  for (int i=0;i<nr[k]-1;i++)
    for (int j=i+1;j<nr[k];j++)
      if (!a[b[k][i]][b[k][j]])
       return 0;
  return 1;
}

void ajoute(int x,int y){
  ut[y]=1; sol[x]=y; nr[y]=0;
  for (int i=0;i<n;i++) a[i][y]=a[y][i]=0;
  for (i=0;i<n;i++)
    for (int j=0;j<nr[i];j++)
      if (b[i][j]==y)
        b[i][j]=b[i][(nr[i]--)-1];
}

void pas_de_solution(void){
  FILE *f=fopen("OUTPUT.TXT","wt");
  fprintf(f,"NO\n");
  fclose(f); exit(1);
}

void determination(void){
  for (int i=0;i<n;i++){
    for (int j=0;j<n;j++)
      if (possible(j)){ajoute(i,j); break;}
    if (sol[i]==-1) pas_de_solution();
  }
}

void solution(void){
  FILE *f=fopen("OUTPUT.TXT","wt");
  fprintf(f,"YES\n%d",sol[0]+1);
  for (int i=1;i<n;i++)
    fprintf(f," %d",sol[i]+1);
  fclose(f);
}

void main(void){
  lecture(); determination(); solution();
}
```

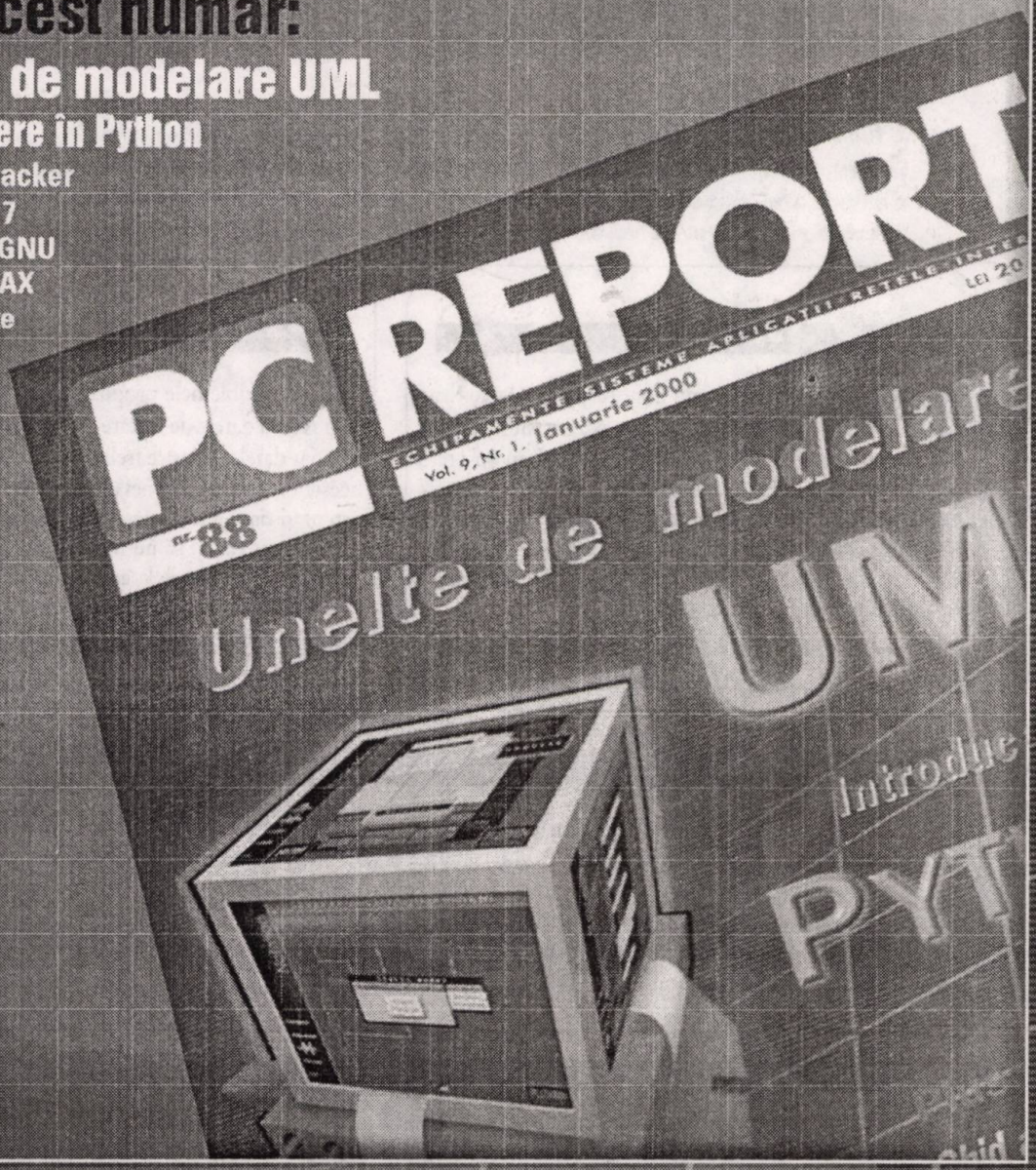
COMPUTER PRESS AGORA

PC REPORT

Vol. 9, Nr. 1, Ianuarie 2000 Lei 20.000

Din acest număr:

- **Unelte de modelare UML**
- **Introducere în Python**
- Ghid anti-hacker
- Solid Edge 7
- Standarde GNU
- XML prin SAX
- PC Concrete



www.pcreport.ro

Revista celor care doresc
să devină **specialiști**
în tehnologia informației și comunicațiilor

COMPUTER PRESS AGORA Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

CONCURS

A luat sfârșit și a doua etapă a concursului organizat de Gazeta de Informatică; în acest număr vă prezentăm clasamentul rundei a doua, clasamentul după două etape, soluțiile problemelor propuse la runda a doua, precum și enunțurile celor trei probleme ale rundei a patra.

Erori...

Numărul cazurilor în care nu erau respectate specificațiile datelor de intrare sau de ieșire a scăzut mult, dar nu a dispărut complet. Vă rugăm să respectați cu strictețe aceste specificații deoarece, datorită faptului că evaluarea este automată, programul de evaluare va considera că soluția este eronată, chiar dacă, din punct de vedere logic, ea este corectă. Nu a dispărut complet nici eroarea de compilare:

```
Error 116: Must be in 8087 mode to compile this
```

Dispozitive I/O

La toate problemele propuse în cadrul concursului Bursele Agora datele de intrare sunt citite dintr-un anumit fișier, iar datele de ieșire trebuie scrise într-un alt fișier. Din aceste motive nu este nevoie să afișați date pe ecran sau să așteptați date introduse de la tastatură. Programatorii în Pascal sunt rugați să nu mai folosească unit-ul Crt deoarece există probleme la execuția programelor; reamintim faptul că evaluarea se face pe un calculator cu procesor Intel Pentium III, având o frecvență de 450 MHz.

Așteptarea reacției

Au existat din nou cazuri în care programele așteptau apăsarea unei taste înaintea terminării execuției. Din punctul de vedere al programului de evaluare, execuția nu se încheie înaintea apăsării acestei taste; acesta va aștepta ca timpul maxim admis pentru execuția testului să expire, după care va întrerupe execuția programului semnalând depășirea timpului admis, iar concurentul nu va primi punctele care i s-ar fi cuvenit pentru testul respectiv.

Termen

Rezolvările problemelor publicate în acest număr pot fi trimise până la data de

20 Februarie 2000

Runda 2 Top 20

1.	Csaba Patcas, Oradea	296	
2.	Adrian Cârcu, Bistrița	288	
3.	Ioana Ileană, Alba Iulia	286	
4.	Cristian Cadar, București	285	
5.	Vlad Vâlceanu, Arad	279	
6.	Mihai Pătrașcu, Craiova	271	
7.	Gabriel Erzse, Oradea	223	
8.	Lucian Burja, Blaj	202	
9.	Silviu Bota, Baia Mare	200	
9.	Daniel Dumitran, București	200	
9.	Ciprian Baicu, Dr. Tn. Severin	200	
9.	Marius Muja, Orăștie	200	
13.	Ileana Constantinescu, Sibiu	195	
13.	Vlad Dascălu, Bacău	195	
15.	Sorin Otescu, Brașov	185	
16.	Pal Szasz, Marghita	180	
17.	Radu Cristian Popescu, București	175	
18.	David Andrei, Râmnicu Vâlcea	171	
19.	Mihai Maerean, Slobozia	155	
19.	Petru Simion Mot , ***	155	

Clasament GInfo Top 50

1	Adrian Cârcu, Bistrița	588
2	Ioana Ileană, Alba Iulia	571
2	Mihai Pătrașcu, Craiova	571
4	Cristian Cadar, București	545
5	Gabriel Erzse, Oradea	523
6	Vlad Vâlceanu, Arad	514
7	Csaba Patcas, Oradea	496
8	Silviu Bota, Baia Mare	488
9	Daniel Dumitran, București	480
10	Lucian Burja, Blaj	430

11.	Cosmin Silvestru Negruseri, Bistrița	420	
12.	Ileana Constantinescu, Sibiu	408	
13.	Adrian Sándor, București	405	
14.	Virgil Șerbănuță, București	400	
15.	Ciprian Baicu, Drobeta Turnu-Severin	388	
16.	Marius Muja, Orăștie	386	
17.	Pal Szasz, Marghita	380	
17.	Marius George Andrei, București	380	
19.	Mihai Stroe, București	370	
20.	Valentin Muste, Baia Mare	368	
21.	Marian Angel, Buftea	356	
22.	David Andrei, Râmnicu Vâlcea	351	
23.	Adrian Bălan, Târgu Jiu	335	
23.	Ionuț Trifan, Buzău	335	
25.	Mihai Maerean, Slobozia	326	
26.	Gabriel Pârvan, Râmnicu Sărat	325	
27.	Marius Neagu, București	313	
28.	Victor Asavei, Drobeta Turnu-Severin	311	
29.	Robert Bădiță, Craiova	309	
30.	Daniel Bolohan, ***	306	
31.	Mihai Dondera, București	305	
32.	George Stelian Pârcăbălescu, Slatina	297	
33.	Horea Coroiu, Cluj	293	
34.	Marius Botiș, Baia Mare	289	
35.	Daniel Popa, Orăștie	288	
36.	Ion Nechita, Slatina	287	
37.	George Alexandru Cocu, Galați	285	
38.	Codruț-Lucian Lazăr, Câmpeni	281	
39.	Vlad Dascălu, Bacău	273	
40.	Andrei Mihnea Răileanu, Slatina	273	
41.	Liviu Andreicuț, București	268	
42.	Paul Rusu, Cluj-Napoca	261	
43.	Radu Cristian Popescu, București	260	
44.	Alexandru Motriuc, Brașov	255	
45.	Ciprian Cană, Vaslui	252	
46.	Cristian Alexandrescu, Botoșani	245	
47.	Mihai Pantiș, Cluj-Napoca	243	
48.	Sorin Otescu, Brașov	241	
49.	Horea-Adrian Haitonic, Năsăud	235	
50.	Mihai Sămărghitan, Târgu Mureș	233	

Runda 2, problema 1

Factorial

(formula lui Stirling)

Această problemă a fost cauza unor nemulțumiri, deoarece se afirma că necesită cunoștințe matematice superioare. Totuși, concurenții care au trimis rezolvări pentru această problemă au obținut un punctaj mediu de 43.80 puncte.

Problema determinării numărului de cifre al lui $n!$ a devenit deja o problemă clasică, ea apărând pentru prima dată la Olimpiada Balcanică de Informatică din 1997, unde a fost propusă de dl. prof. dr. Adrian Atanasiu. O problemă care se reducea la determinarea numărului de cifre a lui $n!$ a fost propusă și în cadrul taberei de pregătire și selecție a Lotului Național de Informatică, organizată la Sinaia în vara acestui an.

În funcție de valoarea maximă a numărului n problema admite diferite soluții. La BOI '97 valoarea maximă a fost 30000, deci o soluție foarte bună s-ar fi putut obține folosind formula:

$$n! = 10^{\lg n!} = 10^{\lg \prod_{i=1}^{n} i} = 10^{\sum_{i=1}^{n} \lg i}$$

deci numărul de cifre poate fi obținut calculând partea întreagă a sumei logaritmilor primelor n numere naturale la care se adună 1. În cadrul concursului nici unul dintre concurenți nu a rezolvat în acest mod problema, dar au existat mulți care au obținut punctajul maxim. Aceștia au simulat calculul valorii numărului $n!$ astfel:

- La fiecare pas (pasul i) se păstrează primele cinci cifre (au existat variante care păstrau până la 9 cifre) și numărul de cifre care urmează.
- Numărul format din cele cinci cifre este înmulțit cu numărul i obținându-se un nou număr.
- Noul număr este împărțit repetat la 10 până când se obține un număr care are cinci cifre; de fiecare dată când se efectuează o împărțire la 10 va crește numărul cifrelor care urmează după grupul primelor cinci.
- În final numărul cifrelor lui $n!$ va fi găsit adunând cinci la grupul cifrelor care urmează după primele cinci cifre.

O descriere în pseudocod a algoritmului este:

```
citește n
gr5=40320; nrcifre=0
pentru i de la 9 la n execută
  gr5←gr5*i
  cât timp gr5>100000 execută
    nrcifre←nrcifre+1; gr5←[gr5/10]
  sfârșit cât timp
sfârșit pentru
afișează nrcifre+5
```

Au rezolvat corect:

Marius George Andrei, București
Liviu Andreicuț, București
Victor Asavei, Dr. Tn. Severin
Ciprian Baicu, Dr. Tn. Severin
Robert Bădiță, Craiova
Silviu Bota, Baia Mare
Marius Botiș, Baia Mare
Adrian Cârcu, Bistrița
Ileana Constantinescu, Sibiu
Horea Coroiu, Cluj
Vlad Dascălu, Bacău
Daniel Dumitran, București
Ioana Ileană, Alba Iulia
Marius Muja, Orăștie
Gabriel Pârvan, Râmnicu Sărat
Mihai Pătrașcu, Craiova
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Radu Cristian Popescu, București
Pal Szasz, Marghita
Virgil Șerbănuță, București
Vlad Vâlceanu, Arad

Observație
Trebuie tratate special cazurile în care *n* este mai mic decât 8.

La tabăra de pregătire de la Sinaia valoarea maximă a lui *n*! era 24572457, deci nici una dintre variantele amintite anterior nu furniza soluția în timp util. Cei care au rezolvat atunci problema, au folosit metoda sumei logaritmilor, puțin modificată. Ei au calculat inițial sumele parțiale pentru numerele până la 24572457 și au păstrat într-un tablou de constante rezultatele obținute pentru numerele care erau multipli de 100000. Apoi, folosind acest tablou au determinat numărul de cifre începând însumarea de la o anumită valoare și adunând apoi cel mult 100000 de termeni; acest algoritm s-a încadrat atunci în timp.

În cadrul concursului Bursele Agora limita impusă a fost 2000000000, deci folosirea algoritmului anterior a fost destul de dificilă, deoarece execuția unui program care calculează suma logaritmilor primelor 2000000000 de numere naturale durează destul de mult. Totuși au existat și concurenți care au adoptat această soluție; Liviu Andreicuț a păstrat valorile pentru multiplii de 200000, Ileană Ioana pentru cei de 250000, Patcas Csaba a ales multiplii de 500000, iar Andrei Marius George pe cei de 1000000.

Majoritatea concurenților au încercat să calculeze suma logaritmilor primelor *n* numere naturale; evident, o astfel de soluție nu respecta limita de timp de o secundă pentru valori mari ale lui *n*. Programele respective funcționau numai pentru valori mici ale lui *n* (cel mult până la 10000000) și au obținut între 20 și 35 de puncte.

Totuși soluția cea mai bună ar fost folosirea formulei lui *Stirling*, prezentată în numărul din decembrie 1999 al Gazetei de Informatică. Vă prezentăm în continuare soluția propusă de *Ileana Constantinescu* din Sibiu:

Listing FACT.CPP

```
/*
Tinem cont de aproximarea lui Stirling care
evalueaza pe n! astfel:
n!=(2*pi*n)^1/2 * (n/e)^n * (1+1/(12*n))
unde pi=3,1415...,
iar e este numarul lui Euler:
e=2,71...
Prin logaritmarea acestei expresii
ajungem la rezultat, deoarece lg(n!)
rotunjit superior reprezinta numarul
de cifre al lui n!
*/

#include <stdio.h>
#include <math.h>

void main(){
  double n,nc=0;
  FILE *pf=fopen("FACT.IN","r");
  fscanf(pf,"%lf",&n);
  fclose(pf);
  if (n<2) nc=1;
  else{
    nc=log10(pow(2*M_PI,0.5))+
            (n+0.5)*log10(n)-n*log10(M_E);
    nc=nc+log10(1+1.0/(12*n));
    nc=ceil(nc);
  }
  pf=fopen("FACT.OUT","w");
  fprintf(pf,"%.0lf",nc);
  fclose(pf);
}
```

Test...

Precizați următorii trei termeni ai șirului:

1 2 4 6 10 12 16 18 22 28 30 36 40 42 ...

Indicați numerele de pe linia punctată!

```
                1
              1 1
              2 1
            1 2 1 1
          1 1 1 2 2 1
          3 1 2 2 1 1
        1 3 1 1 2 2 2 1
.....................
```

Runda 2, problema 3

Submulțimi de sumă dată

Aceasta a fost cea mai dificilă problemă de până acum, media obținută de concurenți fiind de 21.89 puncte. Acest lucru nu este deloc surprinzător deoarece încă nu se cunoaște un algoritm eficient pentru rezolvarea problemei.

Problema submulțimii de sumă dată face parte din clasa problemelor NP-complete, probleme a căror soluție poate fi verificată în timp polinomial, dar pentru care nu se cunoaște un algoritm de rezolvare în timp polinomial. Până în momentul de față cercetătorii nu au reușit să găsească o limită inferioară pentru complexitatea algoritmilor care rezolvă astfel de probleme, deci încă nu s-a demonstrat faptul că există numai algoritmi exponențiali pentru rezolvarea unei astfel de probleme. Teoria NP-completitudinii este foarte interesantă, obținându-se până acum câteva rezultate foarte interesante; totuși ne oprim aici deoarece nu este necesară prezentarea acesteia.

Datorită faptului că încă nu s-a descoperit un algoritm eficient pentru rezolvarea acestei probleme (în cazul general - așa cum a fost propusă spre rezolvare în cadrul concursului GInfo) nimeni nu se aștepta să apară o soluție corectă.

Pentru rezolvarea acceptabilă a unei astfel de probleme trebuie folosiți algoritmi de aproximare (eventual probabiliști) care să determine o soluție cât mai apropiată de cea corectă. Acesta a fost și motivul pentru care redacția a hotărât să acorde și punctaje parțiale pentru această problemă.

Singurul concurent care a obținut punctaj maxim la această problemă a fost Valentin Muste din Baia Mare; programul său a furnizat rezultatul corect pentru cinci dintre cele zece teste și rezultate foarte apropiate în rest, punctajul real fiind de 99.949 puncte. Vă prezentăm în continuare soluția acestui concurent:

Listing SUMA.PAS

```
type ar=array[1..10000] of Real;
     arr=array[1..8200] of Real;
var v,v1:^ar; a,all:^arr;
    m,msol:array[0..10000] of Word;
    rec:array[0..14] of Byte;
    t:Longint absolute 0:$046C;
    t1:Longint; s,su,ssol:Real;
    f:Text; i,j,k,n,maxa,nrm,mmm:Word;

procedure Citire;
begin
  Assign(f,'SUMA.IN'); Reset(f);
  Readln(f,n); Readln(f,s); New(v);
  for i:=1 to n do Readln(f,v^[i]);
  Close(f)
end;

procedure cauta(i,j:Word; var k:Word);
var aux:Real;
begin
  k:=i;
  repeat
    if all^[i]>all^[j] then
    begin
      aux:=all^[i]; all^[i]:=all^[j];
      all^[j]:=aux;
      if k=i then k:=j else k:=i;
    end;
    if k=i then Dec(j) else Inc(i);
  until i=j
end;

procedure Quick(i,j:Word);
var k:Word;
begin
  if i<j then
  begin
    cauta(i,j,k);
    quick(i,k-1);
    quick(k+1,j)
  end
end;
```

CR2P3 Top 10

1. Valentin Muste, Baia Mare	100
2. Gabriel Erzse, Oradea	98
3. Csaba Patcas, Oradea	96
4. Cristian Cadar, București	95
5. Adrian Cârcu, Bistrița	88
6. Ioana Ileană, Alba Iulia	86
7. Lucian Burja, Blaj	82
8. Pal Szasz, Marghita	80
9. Vlad Vâlceanu, Arad	79
10. Mihai Pătrașcu, Craiova	71

```
procedure Restu(var st:Word);
var dr,mij:Word;
begin
  st:=1; dr:=maxa;
  repeat
    mij:=(st+dr) div 2;
    if all^[mij-1]>=s-su
    then dr:=mij else st:=mij
  until st+1=dr;
  if Abs(s-(su+all^[st]))<
                      Abs(s-(su+all^[dr]))
  then dr:=st else st:=dr;
  if Abs(s-(su+all^[st]))<Abs(s-ssol then
  begin
    ssol:=su+all^[st]; msol:=m
  end;
end;

function getel(i:Word):Word;
var k,g:Word;
begin
  k:=1; g:=0;
  repeat k:=k*2; Inc(g); until k>i;
  getel:=g;
end;

procedure reconst(xx:Real);
var i,j:Word; x:Real;
begin
  x:=xx; rec[0]:=0;
  repeat
    for i:=1 to maxa do
      if Abs(a^[i]-x)<0.00000001 then Break;
    j:=getel(i); Inc(rec[0]);
    rec[rec[0]]:=j; x:=x-v^[j];
  until Abs(x)<0.00000001;
end;

procedure worku;
begin
  New(all); Fillchar(all^,Sizeof(all^),0);
  New(a); Fillchar(a^,Sizeof(a^),0);
  maxa:=1; a^[1]:=v^[1];
  for i:=2 to 13 do
  begin
    if n<i then Break;
    for j:=1 to maxa do
      a^[j+maxa]:=a^[j]+v^[i];
    maxa:=2*maxa+1; a^[maxa]:=v^[i];
  end;
  Fillchar(msol,Sizeof(msol),0);
  Fillchar(m,Sizeof(m),0);
  all^:=a^;
  msol[0]:=13;
  quick(1,maxa);
  if n>i then
  begin
    repeat
      nrm:=0; su:=0;
      Fillchar(m,Sizeof(m),0); m[0]:=14;
      repeat
        Inc(nrm); mmm:=Random(Random(40));
        m[nrm]:=m[nrm-1]+1+Random(10);
        if m[nrm]>n then m[nrm]:=n;
        su:=su+v^[m[nrm]];
        if Abs(s-su)<Abs(ssol-s) then
        begin
          ssol:=su; msol:=m; msol[0]:=0
        end;
        restu(j);
      until n=m[nrm];
    until t-t1>175;
  end;
  su:=0; i:=1;
  while msol[i]<>0 do
  begin su:=su+v^[msol[i]]; Inc(i) end;
  if msol[0]<>0 then
    begin
      restu(i);
      reconst(all^[i])
    end
  else begin rec[0]:=0; ssol:=su end
end;

Procedure Scriere;
var ok:Boolean;
begin
  Assign(f,'SUMA.OUT'); Rewrite(f); j:=0;
  while msol[j]<>0 do Inc(j);
  for i:=1 to rec[0] do msol[i+j]:=rec[i];
  j:=j+rec[0];
  repeat
    ok:=true;
    for i:=1 to j-1 do
      if msol[i]>msol[i+1] then
        begin
          k:=msol[i]; msol[i]:=msol[i+1];
          msol[i+1]:=k; ok:=false
        end
  until ok;
  for i:=1 to j do
    if msol[i]<>0 then Write(f,msol[i],' ');
  Close(f);
end;

Begin
  Randomize; t1:=t;
  citire; worku;
  scriere
End.
```

Runda 2, problema 2

Spiriduși

(combinări în hiperspațiu)

Aceasta a fost cea mai ușoară dintre cele șase probleme propuse până acum, media obținută de concurenți fiind de 59.61 puncte. Numărul concurenților care au obținut punctajul maxim a fost destul de redus, mulți obținând numai 95 de puncte.

Majoritatea concurenților au folosit metoda programării dinamice pentru a calcula numărul de puncte din spațiu. Construim o matrice p de dimensiuni $n \times s$ în care $p_{i,j}$ reprezintă numărul de puncte în care poate ajunge GIS într-un hiperspațiu i-dimensional în j zile. Relația de recurență care este foarte ușor de observat este:

$$p_{i,j} = 1 + \sum_{k=1}^{j} p_{i-1,k}$$

Inițial vom completa prima linie și prima coloană a matricei astfel: $p_{1,j}=i$ și $p_{i,1}=1$. Motivul pentru care o mulțime de concurenți au obținut numai 95 de puncte s-a datorat faptului că pentru unul din test numărul de puncte era de ordinul miliardelor (peste 9000000000) și nu putea fi reprezentat pe patru bytes așa cum au considerat ei.

Vă prezentăm în continuare soluția bazată pe această metodă, soluție propusă de *Ileana Constantinescu*.

Listing SPRITES.CPP

```
#include <stdio.h>

long unsigned int p[21][21];
int i,j,n,s;
FILE *pf;

void main(){
  pf=fopen("SPRITES.IN","r");
  fscanf(pf,"%d%d",&n,&s);
  fclose(pf);
  for(i=1;i<=20;i++)p[1][i]=1;
  for(i=1;i<=20;i++)p[i][1]=i;
  for(i=1;i<=20;i++)p[2][i]=i+1;
  for(i=3;i<=20;i++)
    for(j=2;j<=20;j++)
      p[i][j]=p[i-1][j]+p[i][j-1];
```

Au rezolvat corect:

Ciprian Baicu, Drobeta Turnu-Severin
Adrian Bălan, Târgu Jiu
Daniel Bolohan, ***
Silviu Bota, Baia Mare
Mihai Capotă, ***
Adrian Cârcu, Bistrița
Mihai Dondera, București
Daniel Dumitran, București
Gabriel Erzse, Oradea
Ioana Ileană, Alba Iulia
Petru Simion Mot, ***
Marius Muja, Orăștie
Marius Neagu, București
Ion Nechita, Slatina
Cosmin Silvestru Negruseri, Bistrița
Sorin Otescu, Brașov
George Stelian Pârcăbălescu, Slatina
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Andrei Mihnea Răileanu, Slatina
Mihai Sămărghitan, Târgu Mureș
Virgil Șerbănuță, București
Vlad Vâlceanu, Arad
Vlad Veljan, Târgu Mureș

```
  pf=fopen("SPRITES.OUT","w");
  fprintf(pf,"%lu",p[n][s]);
  fclose(pf);
}
```

Problema poate fi rezolvată mult mai simplu, existând o formulă simplă pentru determinarea numărului de puncte din hiperspațiu în care poate ajunge GIS; această formulă este $p=C_{n+s-1}^{n-1}$

Problema se reduce la a găsi numărul partițiilor numărului s formate din exact n elemente (care pot fi și 0). Fiecăruia dintre cele n elemente ale partiției îi corespunde una dintre axele hiperspațiului, iar valoarea elementului respectiv corespunde numărului de unități parcurse de GIS de-a lungul axei corespunzătoare. Formula anterioară poate fi foarte ușor demonstrată folosind metoda inducției matematice. Lăsăm pe seama cititorilor "plăcerea" de a demonstra formula.

O soluție în care a fost folosită această metodă a fost propusă de *Andrei Mihnea Răileanu*; vă prezentăm în continuare această soluție (puțin modificată):

Listing SPRITES2.CPP

```
#include <stdio.h>

long unsigned int p[21][21];
int i,j,n,s;
FILE *pf;

void main(){
  pf=fopen("SPRITES.IN","r");
  fscanf(pf,"%d%d",&n,&s);
  fclose(pf);
  for(i=1;i<=20;i++)p[1][i]=1;
  for(i=1;i<=20;i++)p[i][1]=i;
  for(i=1;i<=20;i++)p[2][i]=i+1;
  for(i=3;i<=20;i++)
    for(j=2;j<=20;j++)
      p[i][j]=p[i-1][j]+p[i][j-1];
  pf=fopen("SPRITES.OUT","w");
  fprintf(pf,"%lu",p[n][s]);
  fclose(pf);
}
```

Runda 4

CR4P1 Labirint

Tom și Jerry se află într-un labirint dreptunghiular de dimensiuni m×n (1≤m, n≤100). Scrieți un program care determină în mai puțin de o secundă lungimea drumului minim dintre pozițiile celor doi, precum și numărul de drumuri de lungime minimă dintre cele două poziții.

Fiecare căsuță din labirint poate fi despărțită de cele patru căsuțe vecine prin ziduri; în cazul în care între două căsuțe vecine nu există un zid, există posibilitatea deplasării din una dintre aceste căsuțe în alta.

Datele de intrare se citesc din fișierul **TOMJERRY.IN** care va conține pe prima linie numerele m și n, separate printr-un singur spațiu, pe a doua linie patru numere întregi (x1 y1 x2 y2) reprezentând pozițiile lui Tom, respectiv Jerry. Pe următoarele m linii se află câte n numere naturale din intervalul [0, 15] care reprezintă codificarea caracteristicilor fiecărei căsuțe.

Fiecăreia dintre cele patru direcții îi corespunde un bit din reprezentarea numărului respectiv (bitul 0 corespunde direcției Nord, bitul 1 direcției Est, bitul 2 direcției Sud, iar bitul 3 direcției Vest).

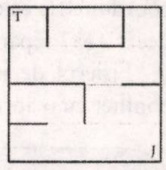
În cazul în care într-o anumită direcție există zid, valoarea bitului corespunzător va fi 1, în caz contrar această valoare va fi 0. De exemplu, pentru o căsuță care are ziduri la Vest și la Nord codificarea va fi $9=1001_2$ (biții corespunzători celor două direcții în care există ziduri au valoarea 1, restul biților au valoarea 0).

Datele de ieșire vor fi scrise în fișierul **TOMJERRY.OUT** care va conține două linii; pe prima linie va apărea lungimea drumului minim dintre pozițiile celor doi, iar pe a doua linie numărul de drumuri de lungime minimă care există între cele două poziții.

Exemplu

Pentru labirintul din figură, fișierul **TOMJERRY.IN** va fi următorul:

```
4 4
1 1 4 4
11 9 7 11
12 4 1 6
13 3 8 3
13 4 4 6
```

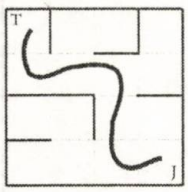


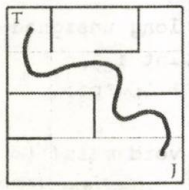
Fișierul **TOMJERRY.OUT** va trebui să fie

```
7
2
```

deoarece Tom trebuie să se deplaseze prin 5 căsuțe intermediare pentru a ajunge la Jerry, deci distanța totală este 7 (sunt incluse și căsuțele în care se află cei doi la început) și

există două drumuri de lungime 7 care pot fi alese, și anume cele prezentate mai jos.





Observații

1. Nu există ieșire din labirint, adică celulele de pe margine vor avea întotdeauna ziduri spre exterior.
2. Există întotdeauna un drum de la poziția lui Tom la cea a lui Jerry.
3. Nu vor apărea contradicții în descrierea căsuțelor labirintului. De exemplu, dacă o căsuță are zid înspre Sud atunci căsuța din Sud va avea obligatoriu zid înspre Nord.

CR4P2 Star Trek

Planeta *GInfo IV* a fost descoperită aproape concomitent de către klingonieni și romulani. Fiecare dintre cele două rase a stabilit mai multe baze pe planetă.

Consiliile Imperiale de pe *Klingon* și *Romulus* au decis că nu își pot permite să pornească un război pentru această planetă și au hotărât să împartă planeta. Fiecare imperiu ar dori să obțină o zonă care să aibă perimetru minim, pentru a fi mai ușor de apărat și care să cuprindă toate bazele stabilite de către acel imperiu pe planetă.

Datorită faptului că bazele au fost stabilite aproape aleator pe planetă neținându-se cont de existența celeilalte rase, cele două zone revendicate de cele două imperii ar putea să aibă porțiuni comune.

Scrieți un program care determină în mai puțin de cinci secunde aria zonei de conflict (zonă revendicată atât de romulani cât și de klingonieni).

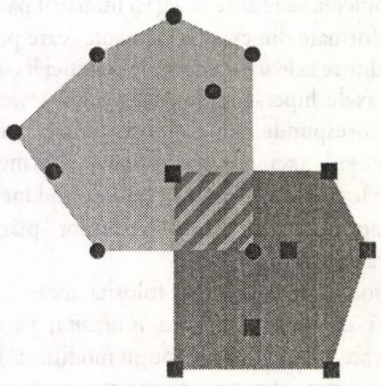
Datele de intrare se citesc din fișierul **STARTREK.IN** care va conține pe prima linie numărul `N` (`1≤N≤100`) al bazelor romulane, pe următoarele `N` linii coordonatele celor `N` baze (două numere naturale mai mici decât `2457` separate printr-un spațiu), pe următoarea linie numărul `M` (`1≤M≤100`) al bazelor klingoniene, pe următoarele `M` linii coordonatele celor `M` baze (două numere naturale mai mici decât `2457` separate printr-un spațiu).

Fișierul de ieșire va trebui să conțină aria zonei de conflict cu o aproximație de `1` miime.

Exemplu

În figură bazele romulane sunt reprezentate prin cerculețe iar cele klingoniene prin pătrățele. Zona revendicată de romulani este cea deschisă la culoare, iar cea revendicată de klingonieni este mai închisă. Zona de conflict este hașurată. Pentru acest exemplu fișierul **STARTREK.IN** va conține:

```
8
1 7
2 6
3 4
3 9
5 10
6 8
7 9
7 4
7
5 6
5 1
7 2
8 4
9 1
9 6
10 4
```



Fișierul de ieșire **STARTREK.OUT** poate conține orice număr din intervalul `[3.999, 4.001]`.

CR4P3 Ordonare

În fișierul **NUMERE.IN** se află cel mult `500000` de numere naturale distincte din intervalul `[0, 999999]`, câte unul pe o linie.

Să se scrie în fișierul **NUMERE.OUT** o listă ordonată crescător a acestor numere, câte un număr pe o linie. Programul va trebui să funcționeze în cel mult cinci secunde.

Exemplu

NUMERE.IN	NUMERE.OUT
2457	8
35	35
87	87
8	2000
2000	2457
8888	7452
9999	8888
7452	9999

Asamblare

Deoarece nu sunteți "cuminți", redacția GInfo a hotărât ca, începând cu runda a patra, să nu mai evalueze rezolvări care conțin cod în limbaj de asamblare.

Descalificări

Au apărut din nou surse aproape identice, următorii concurenți fiind descalificați:

Călin Frîntu, Drobeta Turnu Severin
Alexandru Rapcea, Drobeta Turnu Severin

Introducere în teoria GRAFURILOR

Cristian Cadar

Unul dintre cele mai interesante domenii ale teoriei grafurilor este determinarea de drumuri minime în grafuri. În cadrul acestui episod vă vom prezenta câțiva algoritmi folosiți pentru a determina astfel de drumuri.

Algoritmii de drum minim se împart în patru categorii:
- *sursă unică - destinație unică* (pentru determinarea drumului minim de la un nod numit sursă la un nod numit destinație);
- *sursă unică - destinații multiple* (pentru determinarea drumurilor minime de la un nod numit sursă la toate celelalte noduri ale grafului);
- *surse multiple - destinație unică* (pentru determinarea drumurilor minime de la toate nodurile grafului la un nod numit destinație);
- *surse multiple - destinații multiple* (pentru determinarea drumurilor minime între oricare două noduri ale grafului).

Trebuie menționat faptul că algoritmii de tip *sursă unică - destinație unică* nu au o complexitate mai bună decât cei de tip *sursă unică - destinații multiple*. În ceea ce privește algoritmii de tip *surse multiple - destinație unică*, ei sunt echivalenți cu cei de tip *sursă unică - destinații multiple*, inversând inițial sensul arcelor în cazul grafurilor orientate. Algoritmii de tip *surse multiple - destinații multiple* pot fi implementați folosind algoritmii de tipul *sursă unică - destinații multiple* pentru fiecare nod al grafului dat. Totuși, pentru această problemă există și abordări mai eficiente. Rezumând cele spuse mai sus, rezultă că în esență există două tipuri de algoritmi de drum minim: *sursă unică - destinații multiple* și *surse multiple - destinații multiple*.

O altă clasificare a algoritmilor de drum minim este în funcție de costul muchiilor. Vom avea astfel patru situații:
- graful este neponderat, deci costul pe muchii este unitar;
- graful este ponderat cu toate costurile pozitive;
- graful este ponderat cu costurile pe muchii pozitive și negative, însă fără cicluri negative;
- graful este ponderat cu costurile pe muchii pozitive și negative și conține cicluri negative.

În ceea ce privește prima situație (a unui graf neponderat) lungimea drumului minim dintre două noduri este egală cu numărul minim de muchii dintre cele două noduri. Pentru a afla lungimea drumului minim dintre două noduri în această situație, vom aplica algoritmul *BF* prezentat în articolul anterior, în care vom memora pentru fiecare nod timpul la care a fost vizitat (timpul la care a fost vizitat nodul de start este considerat 0) și predecesorul (tatăl) său din parcurgere.

```
Parcurgere_BF(nod start)
  Coada ← start
  Vizitat(start) ← True
  Timp(start) ← 0
  Tata(start) ← 0
  cât timp coada nu este vidă
    ic ← nodul de la inceputul cozii
    pentru fiecare vecin v nevizitat al lui ic
      Vizitat(v) ← True
      Timp(v) ← Timp(ic)+1
      Tata(v) ← ic
      Adaugă v în coadă
      Avansează în coadă
```

În finalul algoritmului, vom avea în vectorul `Timp` lungimile drumurilor minime de la nodul start la toate celelalte noduri ale grafului.

Reamintim că acest algoritm are complexitatea $O(n+m)$. Pentru a reconstitui drumul minim de la nodul `start` la un anumit nod `k`, se folosește bineînțeles vectorul tăților:

```
Afiseare_Drum(nod k)
  dacă Tata(k) ≠0
    atunci Afiseare_Drum(Tata(k))
  scrie k
```

De remarcat faptul că fiecare drum minim între două puncte ale unui graf conține în interiorul său alte drumuri minime. Mai precis, dacă $i_1, i_2, ..., i_n$ este un drum minim și i_k un nod aparținând acestui drum, atunci $i_1, i_2, ..., i_k$ și $i_k, i_{k+1}, ..., i_n$ sunt drumuri minime. Demonstrația acestui fapt este evidentă și stă la baza majorității algoritmilor de drum minim existenți.

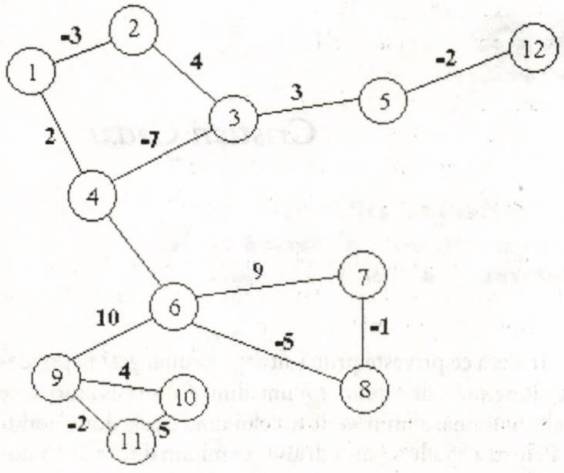
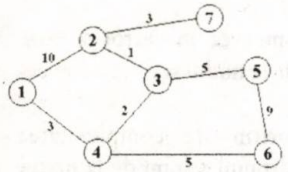


Figura 1 - Graf care conține un ciclu negativ 1-2-3-4-1 de cost -4

În ceea ce privește grafurile care conțin cicluri de cost negativ, cum este cazul grafului din figura 1, drumul minim dintre oricare două noduri este $-\infty$. Acest lucru este evident, deoarece oricare drum dintre două noduri poate fi micșorat prin parcurgerea repetată a ciclului negativ.

Cel mai cunoscut, dar și unul dintre cei mai eficienți algoritmi de tip sursă unică - destinații multiple pentru grafuri cu muchii de cost pozitiv este algoritmul *Dijkstra*. În cele ce urmează, vom prezenta și vom demonstra algoritmul Dijkstra.

Acest algoritm află toate drumurile minime de la un nod s (sursă) la toate celelalte noduri ale grafului în ordine crescătoare a lungimii acestor drumuri. De exemplu, pentru graful din figura 2, în care sursa este considerată nodul 1, drumurile sunt generate în ordinea:



Drum	Cost
1 – 4	3
1 – 4 – 3	5
1 – 4 – 3 – 2	6
1 – 4 – 6	8
1 – 4 – 3 – 2 – 7	9
1 – 4 – 3 – 5	10

Figura 2

În cazul în care două drumuri au același cost, algoritmul Dijkstra le poate genera în orice ordine, în funcție de modul în care au fost memorate muchiile grafului.

Algoritmul lui Dijkstra folosește o mulțime S care reține nodurile pentru care (la un anumit moment) s-au generat drumurile minime de la sursă la nodurile respective. Mulțimea S este inițializată cu nodul sursă. Apoi, aceasta este completată rând pe rând cu toate celelalte noduri ale grafului, în ordine crescătoare față de lungimea drumului minim de la s (nodul sursă) la acele noduri. Pentru graful din figura 2, ordinea în care sunt introduse nodurile în S este: 1, 4, 3, 2, 6, 7, 5.

De asemenea, algoritmul folosește un tablou unidimensional *DIST*, *DIST*(i) reținând lungimea drumului minim (distanța) de la s la i. La începutul algoritmului, vectorul *DIST* este inițializat cu lungimile muchiilor de la s la nodurile respective. În cazul în care nu există muchie între nodul s și un nod k, *DIST*(k) va fi inițializat cu ∞. *DIST*(s) va fi inițializat cu valoarea 0.

Algoritmul *Dijkstra* are n-1 pași. La fiecare pas, este introdus în S un nod care bineînțeles nu aparține mulțimii și pentru care distanța de la s la acesta este minimă (deci acel $k \notin S$ pentru care *DIST*(k) este minim).

Având aceste structuri, putem face următoarele observații care prezintă și demonstrează modul în care acționează algoritmul Dijkstra.

1. În momentul în care se generează un nou drum minim, atunci acel drum începe din s, se termină într-un nod v care nu aparține lui S și conține numai noduri ce aparțin lui S. Vom demonstra în cele ce urmează că toate nodurile de pe drumul minim (s, ..., v) aparțin lui S. Pentru aceasta, vom presupune prin absurd că există un nod w care aparține acestui drum și care nu aparține lui S. Atunci drumul (s, ..., w, ..., v) poate fi împărțit în două drumuri minime (s, ..., w) și (w, ..., v). Dar lungimea drumului (s, ..., w) este mai mică decât cea a drumului (s, ..., w, ..., v). Rezultă contradicție cu faptul că nodurile sunt introduse în S în ordine crescătoare a distanței față de s. Deci, în *DIST*(k) vom reține la un moment dat lungimea drumului minim de la s la k, trecând doar prin noduri din S.
2. La un anumit moment, după generarea unui drum minim pentru un nod v, acesta este introdus în mulțimea S, deci lungimea unui drum minim de la nodul s la un nod w care nu aparține lui S și cu toate celelalte noduri intermediare în S se poate modifica (mai precis se poate micșora). Dacă acest drum se schimbă, atunci înseamnă că există un drum mai scurt de la s la nodul respectiv, care trece prin ultimul nod introdus în S, nodul v. Atunci, așa cum am demonstrat anterior, toate nodurile intermediare care aparțin drumului (s, ..., v, ..., w) trebuie să aparțină lui S. Drumul (s, ..., v, ..., w) poate fi descompus (așa cum am arătat anterior) în două drumuri minime (s, ..., v) și (v, ..., w). Bineînțeles, drumul (v, ..., w) poate fi ales chiar muchia (v, w), iar lungimea drumului (s, ..., v) va fi reținută în *DIST*(v). Este corect să alegem muchia (v, w) deoarece pe parcursul algoritmului se va încerca optimizarea lungimii drumului minim de la s la w prin toate nodurile v din S.

Pe lângă demonstrarea corectitudinii algoritmului, aceste observații oferă indicațiile necesare implementării algoritmului Dijkstra:

```
Dijkstra(graf G,nod s)
  Inițializare S,DIST
  pentru pas=1,n-1
    selectează nodul v∉S pentru care
                              DIST(v) este minim
    S=S+{v}
    pentru w∉S și există muchia (v,w)
      dacă DIST(w)>DIST(v)+cost(v,w)
        atunci DIST(w)=DIST(v)+cost(v,w)
```

Se observă că modul în care este implementată mulțimea *S* este decisiv pentru o bună complexitate a algoritmului Dijkstra. Dacă mulțimea *S* este un vector cu elemente de tip logic, atunci complexitatea algoritmului este $O(n^2+m) = O(n^2)$, găsirea minimului într-o astfel de structură fiind $O(n)$ (linia (1)), iar introducerea unui element (linia (2)) și stabilirea apartenenței (linia (3)) $O(1)$. Pentru a reduce complexitatea algoritmului, vom reține pe lângă vectorul logic *S*, mulțimea *V/S* sub forma unui heap. Se știe că structura de heap permite:

- găsirea minimului în $O(1)$;
- introducerea unui element în $O(\log n)$;
- ștergerea unui element în $O(\log n)$;
- căutarea unui element în $O(n)$.

Folosind doar această structură, evaluarea liniei (1) se va face în $O(n)^3$, a liniei (2) în $O(n \log n)$, a liniei (3) în $O(n \cdot m)$, iar a liniei (4) în $O(m \log n)$. Folosind însă și vectorul logic *S*, vom reuși să reducem linia (3) la complexitatea $O(m)$, deci complexitatea totală a algoritmului Dijkstra devine $O(m \log n)$. Menționăm că folosirea heap-urilor de tip Fibonacci reduc complexitatea la $O(n \log n+m)$, complexitate optimă pentru acest algoritm.

Dacă graful conține muchii de cost negativ, algoritmul Dijkstra nu mai funcționează corect, deoarece nu putem găsi - ca și până acum - nodurile cele mai apropiate de sursă, în ordine crescătoare a distanței față de aceasta. Să presupunem prin reducere la absurd că putem determina și în acest caz mulțimea *S* folosită de algoritmul Dijkstra. Fie v ultimul nod introdus în mulțimea *S*, deci un nod pentru care cunoaștem drumul minim de la s la el, drum ce conține numai noduri din *S*. Fie w un nod care nu aparține lui *S* și (v, w) o muchie de cost negativ foarte mic. Atunci drumul minim de la s la v va fi compus din drumul minim de la s la w urmat de muchia (w,v). Obținem astfel contradicție cu faptul că vârfurile de pe drumul minim de la s la v aparțin lui *S*, deci nu putem reține - ca în cazul algoritmului Dijkstra - această mulțime *S*.

Această problemă este rezolvată de algoritmul *Bellman-Ford*, care în plus determină și existența ciclurilor de cost negativ care pot fi atinse pornind din nodul sursă. Ca și în cazul algoritmului Dijkstra, vom folosi vectorul *DIST* care reține distanța minimă găsită la un moment dat de la s la celelalte noduri. De asemenea, algoritmul folosește o coadă *Q*, care este inițializată cu nodul s. Apoi, la fiecare pas, algoritmul scoate un nod v din coadă și găsește toate nodurile w a căror distanță de la sursă la acestea poate fi optimizată prin folosirea nodului v. Dacă nodul w nu se află deja în coadă, el este adăugat acesteia. Acești pași se repetă până când coada devine vidă. În continuare este prezentat algoritmul Bellman-Ford în pseudocod.

```
Bellman-Ford(graf G, nod s)
Inițializare Q,DIST
cât timp Q nu este vidă
  scoate v din coadă
  pentru fiecare muchie (v,w)
    dacă DIST(w)>DIST(v)+cost(v,w)
      atunci DIST(w)=DIST(v)+cost(v,w)
    dacă w∉Q atunci inserează w în Q
```

Observăm că fiecare nod poate fi scos din coadă de cel mult n ori, deci complexitatea algoritmului este $O(n \cdot m)$. Așa cum este prezentat algoritmul, el va rula la infinit în cazul ciclurilor de cost negativ. Această problemă se rezolvă foarte ușor, deoarece în momentul în care un nod a fost scos pentru a $n+1$ oară din coadă, atunci în mod sigur există un ciclu negativ în graful dat.

O situație particulară deosebit de importantă a algoritmului Dijkstra este cazul grafurilor orientate aciclice, având costuri pozitive. În acest caz, complexitatea algoritmului Dijkstra va fi redusă la $O(n+m)$. Pentru aceasta, nodurile vor fi parcurse în ordine topologică. Astfel linia (1) va fi eliminată, ordinea de parcurgere a nodurilor fiind cunoscută dinainte. Corectitudinea acestei strategii este evidentă, deoarece din s nu vom putea ajunge la un nod i înainte de a ajunge la un nod k, dacă nodul i se află după nodul k în sortarea topologică (deci există un drum de la nodul k la nodul i, dar nu și invers).

În cele ce urmează, ne vom ocupa de algoritmi de tipul surse multiple - destinații multiple. O primă posibilitate de aflare a drumurilor minime între oricare două perechi de vârfuri ale unui graf este aplicarea de n ori a algoritmului Dijkstra, complexitatea astfel obținută fiind $O(n^3)$, pentru cazul în care mulțimea *S* este implementată sub formă de vector de tip logic. În cele ce urmează, vom prezenta însă și un alt algoritm care realizează acest lucru, și care, deși are aceeași complexitate, este mult mai simplu de implementat. Acest algoritm este algoritmul lui *Floyd* și se bazează pe *principiul programării dinamice*. Algoritmul folosește o matrice tridimensională D, $D(i, j, k)$ reținând lungimea drumului minim de la i la j, folosindu-se în componența drumurilor doar primele k noduri. Prin definiție, $D(0, i, j) = A(i, j)$, unde

$$A(i,j)=\begin{cases} cost(i,j), \text{ dacă există muchie între vârful } i \text{ și vârful } j \\ \infty, \text{ în caz contrar} \end{cases}$$

Bineînțeles, lungimile drumurilor minime între două vârfuri se vor afla în matricea bidimensională $D(n)$.

Se poate determina ușor relația de recurență pentru a completa matricea D:

$D(k, i, j) = min(D(k\text{-}1, i, j), D(k\text{-}1, i, k) + D(k\text{-}1, k, j))$

Se observă că pentru a completa matricea $D(k)$ avem nevoie doar de matricea $D(k\text{-}1)$, deci în total vom avea nevoie doar de două matrice bidimensionale. Mai mult chiar, observând că $D(k, i, k) = D(k\text{-}1, i, k)$ și că $D(k, k, j) = D(k\text{-}1, k, j)$, va rezulta că nu vom avea nevoie decât de o singură matrice. Egalitățile enunțate mai sus sunt evident corecte, deoarece - cu excepția cazului în care există un ciclu de cost negativ - nu putem optimiza un drum care se termină și care începe cu k folosind tot nodul k.

Rezumând cele spuse mai sus, rezultă algoritmul foarte concis de mai jos:

```
Floyd(graf G)
  D ← A
  pentru k=1,n
    pentru i=1,n
      pentru j=1,n
        dacă D(i,j)>D(i,k)+D(k,j)
          atunci D(i,j)=D(i,k)+D(k,j)
```

Menționăm că algoritmul lui Floyd funcționează corect pentru grafuri cu costuri negative pe muchii.

Având la bază acest algoritm este ușor să realizăm un algoritm care transformă matricea de adiacență în matricea drumurilor (matricea drumurilor este o matrice binară pentru care elementul (i, j) este 1 doar dacă există drum între vârfurile i și j). Acest algoritm se numește algoritmul lui *Roy-Warshall*.

```
Roy-Warshall(graf G)
  D ← A // A este matricea de adiacenţă
    pentru k=1,n
      pentru i=1,n
        pentru j=1,n
          dacă D(i,k)=1 şi D(k,j)=1
            atunci D(i,j)=1
```

În algoritmii prezentați în acest articol nu am precizat cum se face reconstituirea drumurilor minime. Am omis intenționat acest lucru, deoarece este foarte simplu de realizat, prin folosirea unui vector de predecesori, așa cum am arătat în cazul algoritmului *BF* de generare a drumurilor minime într-un graf neponderat.

Cristian Cadar este student în anul II la Universitatea Politehnică București. Poate fi contactat prin e-mail la adresa ccadar@pcnet.pcnet.ro

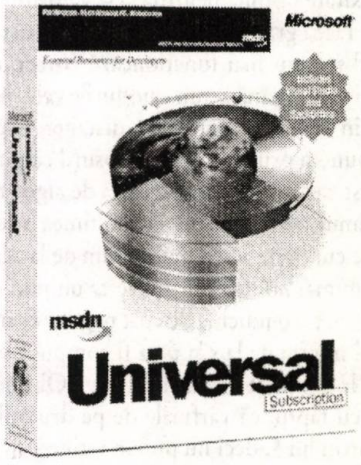
PREMIUL LUNII

Microsoft România oferă și în această lună abonaților 3 pachete MSDN Universal Subscription

MSDNTM Subscriptions vă ajută să beneficiați de avantajele celor mai recente instrumente și tehnologii Microsoft. Cu sistem de abonare flexibil, organizat pe mai multe nivele și avantajul de fi actualizat la timp pe tot parcursul anului, MSDN Subscription este esențial pentru orice dezvoltator.

MSDN Universal Subscription

MSDN Universal Subscription face mai ușoară dezvoltarea unor soluții particulare oferind toate informațiile, tehnologia și instrumentele de dezvoltare de care aveți nevoie. Este pur și simplu modul cel mai eficient, și ieftin de a dezvolta și testa soluții.



Prin flexibilitatea unor nivele de abonament adaptate fiecăruia și a avantajului unor actualizări la timp de-a lungul întregului an, un abonament MSDN este esențial pentru orice dezvoltator.

Abonamentul de tip Universal include toate facilitățile unuia de tip Professional plus toate produsele apărute de-a lungul anului, incluzând: Microsoft Visual Studio 6.0 Enterprise Edition, BackOffice Test Platform, instrumente de dezvoltare suplimentare, versiuni internaționale, produse și instrumente în curs de apariție, actualizări și produse nou-lansate.

Câștigătorii din luna decembrie sunt:

1. Marinescu Sorin Theodor
str. A.I. Cuza nr. 5, bl. Romarta, sc. A, ap. 28
1100 - CRAIOVA

2. Luca Sergiu
str. Saturn nr. 20, bl. C6, ap. 19,
tel. 036/439245
6200 - GALAȚI

3. Ciobaca Ștefan
str. Dornelor nr. 15, bl. B3, sc. C, ap. 7
5800 - Suceava

Pagini WEB cu JavaScript

Mihai Scorțaru

În cadrul acestui episod vom prezenta o modalitate de accesare a paginilor vizitate anterior, vom descrie posibilitățile de manipulare a listelor de selecție; de asemenea vom prezenta funcțiile sistem, precum și numeroase alte proprietăți specifice formularelor, câmpurilor cuprinse în acestea sau a întregului document.

Obiectul history

După ce ați devenit familiari cu interfața unui anumit program de navigare, ați utilizat foarte des butoanele ***Back***, ***Forward***, ***Home*** etc. Utilitatea acestor butoane este determinată de posibilitatea programului de navigare de a memora ultimele locații vizitate în timpul sesiunii curente. Lista acestor locații poartă denumirea de ***history list*** și poate fi accesată prin intermediul obiectului `history`.

Singura proprietatea a obiectului `history` este `length` și indică numărul de intrări (URL-uri vizitate) corespunzătoare obiectului `history` atașat ferestrei curente. Dacă este creată o nouă fereastră prin intermediul unor instrucțiuni *JavaScript* (metoda `window.open()`) sau folosind facilitățile oferite de programul de navigare (comanda ***New*** din meniul ***File***), noua fereastră nu va păstra informații referitoare la locațiile vizitate în cadrul ferestrei părinte, dar va avea propria listă de locații care va începe, eventual, cu URL-ul care este încărcat implicit la deschiderea unei noi ferestre.

Pentru utilizator este destul de convenabil să parcurgă locațiile vizitate recent prin intermediul butoanelor oferite de programul de navigare, dar pentru ca acest lucru să fie realizat automat (de către proiectantul paginii) trebuie folosite anumite metode ale obiectului `history`. Denumirile acestor metode sunt identice cu denumirile unelteltor puse la dispoziție de programul de navigare pentru scopuri similare și anume: `go()`, `back()` și `forward()`.

Metoda go()

În cadrul programelor de navigare lista celor mai recente locații vizitate apare sub forma unui meniu, primele zece intrări putând fi accesate și prin intermediul tastelor `0`, `1`, ...,`9`. Numai aceste ultime zece locații sunt accesibile prin intermediul tastaturii; pentru a alege o altă locație, aceasta trebuie selectată cu ajutorul mouse-ului. Metoda `go()` ne permite să "sărim" la o anumită locație vizitată recent și are sintaxa `history.go(target)`, unde `target` poate fi un număr întreg sau un șir de caractere. În cazul în care valoarea acestui parametru este un număr întreg, acesta corespunde poziției URL-ului care va fi încărcat față de URL-ul curent; valoarea `0` duce la reîncărcarea paginii curente. Un număr pozitiv sau negativ duce la încărcarea paginii Web corespunzătoare unui URL aflat în lista locațiilor vizitate. În cazul în care vom avea ca parametru un șir de caractere, acesta va trebui să conțină URL-ul (integral sau parțial) paginii care va fi încărcată. Când parametrul `target` al metodei `go()` este un șir de caractere, va fi încărcată pagina corespunzătoare primului URL care conține respectivul șir în interiorul său.

Metoda back()

Această metodă a fost introdusă pentru a permite întoarcerea la pagina vizitată anterior. Sintaxa acestei metode este `history.back()` și este echivalentă cu `history.go(-1)`.

În cazul în care nu mai există nici o pagină vizitată anterior, metoda `back()` nu are nici un efect.

Metoda forward()

Această metodă a fost introdusă pentru a permite încărcarea paginii corespunzătoare următorului URL din lista locațiilor vizitate recent.

Sintaxa acestei metode este `history.forward()` și este echivalentă cu `history.go(1)`.

În cazul în care am ajuns la primul element din lista locațiilor vizitate recent, metoda `forward()` nu are nici un efect.

Manipularea listelor de selecție

Listele de selecție permit standardizarea valorilor pe care utilizatorul le poate transmite prin intermediul unui formular. Așa cum am spus în unul din episoadele anterioare o listă de selecție este creată cu ajutorul marcajului **<SELECT>**, iar fiecare element (opțiune) al listei poate fi

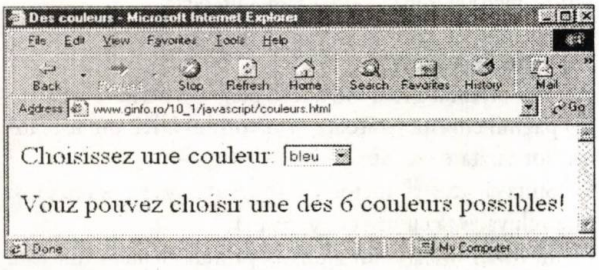
creat prin intermediul marcajului **<OPTION>**. Pentru a avea acces la o listă de selecție din interiorul unei secvențe de instrucțiuni *JavaScript* va trebui să apară și parametrul **name** al marcajului **<SELECT>**, valoarea acestuia reprezentând identificatorul obiectului care reprezintă lista de selecție.

Proprietatea length

Obiectele care reprezintă liste de selecție au o proprietate numită **length** prin intermediul căreia poate fi obținut numărul opțiunilor din listă. Următorul document HTML ilustrează modul de utilizare a acestei proprietăți:

```
<HTML>
  <HEAD><TITLE> Des couleurs </TITLE></HEAD>
  <BODY>
    <BASEFONT size="5">
    <FORM>
      Choisissez une couleur:
      <SELECT name="colors">
        <OPTION value="blue">bleu
        <OPTION value="yellow">jaune
        <OPTION value="red">rouge
        <OPTION value="green">vert
        <OPTION value="black">noir
        <OPTION value="white">blanc
      </SELECT>
    </FORM>
    Vouz pouvez choisir une des
    <SCRIPT language="JavaScript">
      document.write(document.forms[0].
                              colors.length);
    </SCRIPT>
    couleurs possibles!
  </BODY>
</HTML>
```

În urma încărcării paginii anterioare, fereastra programului de navigare ar putea arăta astfel:

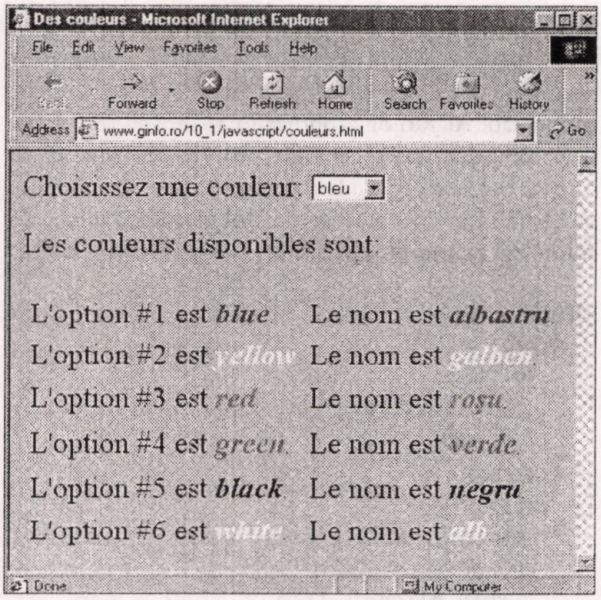


Proprietatea options

Opțiunile prezente în listă sunt disponibile prin intermediul proprietății `options`, care este, de fapt, un vector care conține elementele listei. Următorul exemplu ilustrează modul de utilizare a acestei proprietăți:

```
<HTML>
  <HEAD><TITLE> Des couleurs </TITLE></HEAD>
  <BODY bgcolor="#cccccc">
    <BASEFONT size=5>
    <FORM name=colors>
      Choisissez une couleur:
      <SELECT name="colors">
        <OPTION name="albastru"
                              value="blue">bleu
        <OPTION name="galben"
                             value="yellow">jaune
        <OPTION name="roşu"
                              value="red">rouge
        <OPTION name="verde"
                             value="green">vert
        <OPTION name="negru"
                             value="black">noir
        <OPTION name="alb"
                            value="white">blanc
      </SELECT>
    </FORM>
    Les couleurs disponibles sont:
    <BR>
    <BR>
    <TABLE cellspacing="5">
    <SCRIPT language="JavaScript">
      for (i=0;i<document.colors.
                          colors.length;i++){
        document.write("<TR><TD>");
        document.write("<FONT size=5>");
        document.write("L'option #");
        document.write(i+1);
        document.write(" est ");
        document.write("<FONT color=");
        document.write(document.colors.
                  colors.options[i].value);
        document.write("><B><I>");
        document.write(document.colors.
                  colors.options[i].value);
        document.write("</B></I>");
        document.write("</FONT></FONT>.");
        document.write("<TD><FONT size=5>");
        document.write("Le nom est ");
        document.write("<B><I>");
        document.write("<FONT color=");
        document.write(document.colors.
                  colors.options[i].value);
        document.write(">");
        document.write(document.colors.
                  colors.options[i].name);
        document.write("</B></I>");
        document.write("</FONT></FONT>.");
      }
    </SCRIPT>
    </TABLE>
  </BODY>
</HTML>
```

Fereastra programului de navigare ar putea afișa următoarea pagină Web:

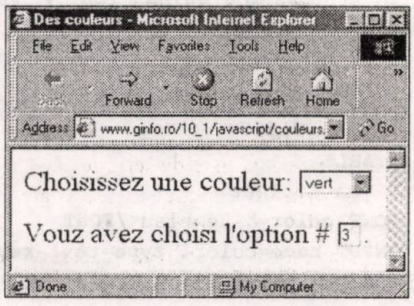


Proprietatea selectedIndex

Pentru a determina indicele opțiunii selectate putem folosi proprietatea `selectedIndex` (numerotarea indicilor începe cu 0). În cazul în care avem o listă care permite selectarea mai multor opțiuni, proprietatea `selectedIndex` va indica indicele primei opțiuni selectate. Următorul document HTML determină apariția unei pagini Web în care vom putea alege o anumită culoare, iar indicele culorii selectate va apărea într-o casetă.

```
<HTML>
  <HEAD>
    <TITLE>
       Des couleurs
    </TITLE>
  </HEAD>
  <BODY>
    <FORM name=colors>
      <BASEFONT size="5">
      Choisissez une couleur:
      <SELECT name="colors"
              onchange="index.value=
                        colors.selectedIndex">
        <OPTION value="blue">bleu
        <OPTION value="yellow">jaune
        <OPTION value="red">rouge
        <OPTION value="green">vert
        <OPTION value="black">noir
        <OPTION value="white">blanc
      </SELECT>
      <P>
      Vouz avez choisi l'option #
      <INPUT name=index type=text readonly
                              size=1 value=0>.
    </FORM>
  </BODY>
</HTML>
```

La un moment dat, fereastra programului de navigare ar putea arăta astfel:

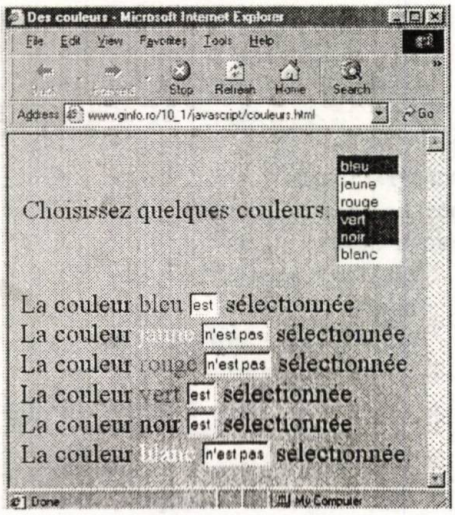


Dacă dorim să determinăm toate opțiunile selectate într-o listă care permite selecții multiple putem folosi proprietatea `selected` a fiecărui element al vectorului `options` asociat listei de selecție. Următorul document HTML determină afișarea unei pagini Web care permite selectarea mai multor culori și afișează care dintre culori sunt selectate.

```
<HTML>
  <HEAD><TITLE> Des couleurs </TITLE>
  <SCRIPT language="JavaScript">
    function value(i){
      return (document.colors.colors.
      options[i].selected)?"est":"n'est pas";
    }
    function size(i){
      return (document.colors.colors.
                    options[i].selected)?2:7;
    }
    function index(){
      document.colors.color1.value=value(0);
      document.colors.color1.size=size(0);
      document.colors.color2.value=value(1);
      document.colors.color2.size=size(1);
      document.colors.color3.value=value(2);
      document.colors.color3.size=size(2);
      document.colors.color4.value=value(3);
      document.colors.color4.size=size(3);
      document.colors.color5.value=value(4);
      document.colors.color5.size=size(4);
      document.colors.color6.value=value(5);
      document.colors.color6.size=size(5);
    }
  </SCRIPT></HEAD>
  <BODY bgcolor="#cccccc">
    <FORM name=colors>
      <BASEFONT size="5">
      <TABLE>
        <TR valign="middle"><TD>
          <FONT size=5>
            Choisissez quelques couleurs:
          </FONT>
        <TD><SELECT size=6  onchange="index()"
                        multiple name="colors">
```

```
            <OPTION value="blue">bleu
            <OPTION value="yellow">jaune
            <OPTION value="red">rouge
            <OPTION value="green">vert
            <OPTION value="black">noir
            <OPTION value="white">blanc
          </SELECT>
        </TABLE>
        <P> La couleur
        <FONT color=blue>bleu</FONT>
        <INPUT name=color1 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
        La couleur
        <FONT color=yellow>jaune</FONT>
        <INPUT name=color2 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
        La couleur <FONT color=red>rouge</FONT>
        <INPUT name=color3 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
        La couleur
        <FONT color=green>vert</FONT>
        <INPUT name=color4 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
        La couleur
        <FONT color=black>noir</FONT>
        <INPUT name=color5 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
        La couleur
        <FONT color=white>blanc</FONT>
        <INPUT name=color6 type=text readonly>
        s&eacute;lectionn&eacute;e.<BR>
      </FORM>
      <SCRIPT language="JavaScript">
        index();
      </SCRIPT>
    </BODY>
</HTML>
```

Dacă vom selecționa prima, a patra și a cincea culoare, pagina Web ar putea arăta astfel:



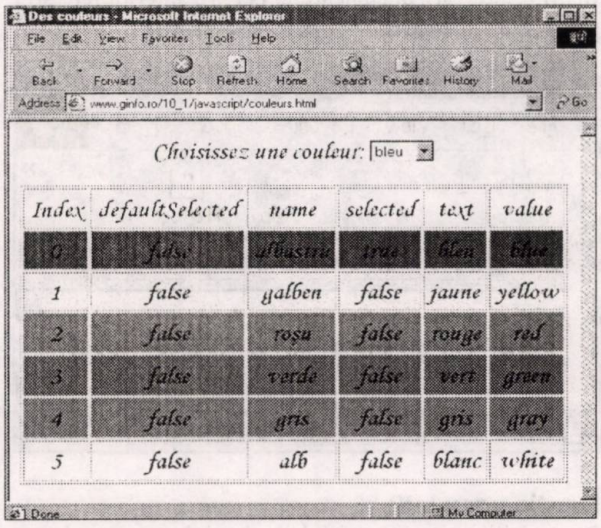
Proprietăți ale elementelor vectorului options

După cum reiese și din cele prezentate anterior, elementele vectorului `options` sunt și ele obiecte și au anumite proprietăți. Alături de proprietatea `selected` care a fost deja amintită, există și alte proprietăți cum ar fi: `defaultSelected`, `index`, `name`, `selected`, `text` și `value`. Următorul document HTML ilustrează modul de folosire al acestor proprietăți:

```
<HTML>
  <HEAD>
    <TITLE>
      Des couleurs
    </TITLE>
  </HEAD>
  <BODY>
    <BASEFONT size=5 face="Monotype Corsiva">
    <FORM name=colors>
      <DIV align=center>
        Choisissez une couleur:
        <SELECT name=colors>
        <OPTION name=albastru value=blue>bleu
          <OPTION name=galben
                              value=yellow>jaune
          <OPTION name=roșu value=red>rouge
          <OPTION name=verde
                              value=green>vert
          <OPTION name=gris value=gray>gris
          <OPTION name=alb value=white>blanc
        </SELECT>
      </DIV>
      <P>
      <FONT face="Monotype Corsiva">
      <TABLE border=1 cellpadding=5>
        <TR align="center">
          <TD><FONT size=5>Index
          <TD><FONT size=5>defaultSelected
          <TD><FONT size=5>name
          <TD><FONT size=5>selected
          <TD><FONT size=5>text
          <TD><FONT size=5>value
        <SCRIPT language="JavaScript>
          for (i=0;i<6;i++){
            document.write("<TR align="+
            "center bgcolor="+colors.colors.
                        options[i].value+">");
            document.write("<TD>");
            document.write("<FONT size=5>");
            document.write(colors.colors.
                        options[i].index);
            document.write("<TD>");
            document.write("<FONT size=5>");
            document.write(colors.colors.
                options[i].defaultSelected);
            document.write("<TD>");
```

```
            document.write("<FONT size=5>");
            document.write(colors.colors.
                           options[i].name);
            document.write("<TD>");
            document.write("<FONT size=5>");
            document.write(colors.colors.
                         options[i].selected);
            document.write("<TD>");
            document.write("<FONT size=5>");
            document.write(colors.colors.
                           options[i].text);
            document.write("<TD>");
            document.write("<FONT size=5>");
            document.write(colors.colors.
                          options[i].value);
        }
      </SCRIPT>
    </TABLE>
  </FORM>
 </BODY>
</HTML>
```

În urma încărcării acestei pagini fereastra programului de navigare ar putea arăta astfel:



Metode specifice câmpurilor

Obiectele corespunzătoare câmpurilor din formulare partajează un set de metode care permit determinarea momentului în care valorile lor se modifică. Aceste metode sunt `blur()`, `focus()` și `select()`.

Metoda blur()

Metoda `blur()` determină pierderea controlului de către câmpul corespunzător obiectului pentru care este apelată. Aceasta duce la executarea instrucțiunilor corespunzătoare atributului **onblur** și, eventual, a celor corespunzătoare atributului **onchange**, în cazul în care valoarea a fost schimbată; sintaxa metodei este `denumire_câmp.blur()`.

Metoda focus()

Metoda pereche pentru `blur()` este `focus()`; aceasta determină primirea controlului de către câmpul corespunzător obiectului pentru care este apelată, fapt ce duce la executarea instrucțiunilor corespunzătoare atributului **onfocus**; sintaxa metodei este `denumire_câmp.focus()`.

Metoda select()

Metoda `select()` are ca efect selectarea textului introdus în zona corespunzătoare obiectului pentru care a fost apelată. Se execută instrucțiunile corespunzătoare atributului **onselect**; sintaxa metodei este `denumire_câmp.select()`.

Funcții sistem

Există funcții *JavaScript* care nu se referă la un anumit obiect, ele putând fi folosite în orice situație; câteva dintre aceste funcții sunt `escape()`, `eval()`, `parseFloat()`, `parseInt()` și `unescape()`.

Funcția escape()

Fiecărui caracter afișat pe ecran îi corespunde un anumit cod unic; alături de codurile corespunzătoare caracterelor "normale" (cifre, litere, diferite simboluri) există și coduri care au funcții speciale (caractere de sfârșit de linie, simboluri speciale etc.). Aceste coduri corespunzătoare caracterelor poartă denumirea de *coduri escape*; există două moduri mai des folosite pentru reprezentarea codurilor escape și anume *ASCII* (*American Standard Code for Information Interchange*) și *EBCDIC* (*Extended Binary-Coded Decimal Interchange Code*).

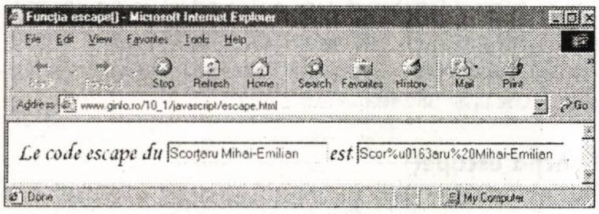
Limbajul *JavaScript* folosește o reprezentare a codului ASCII care cuprinde caracterul '%' urmat de codul ASCII corespunzător caracterului. Funcția `escape()` returnează, de obicei, reprezentarea numerică a unui șir de caractere transmis ca parametru.

În cazul anumitor caractere în șirul returnat de funcția `escape()` nu vor apărea codurile caracterelor respective, acestea fiind înlocuite de caracterele propriu-zise. Următorul document HTML ilustrează modul de utilizare a funcției `escape()`.

```
<HTML>
  <HEAD>
    <TITLE> Funcţia escape() </TITLE>
    <SCRIPT language=JavaScript>
      function modify(){
        document.escape.cod.value=
          escape(document.escape.text.value);
        document.escape.cod.size=
            document.escape.cod.value.length;
        document.escape.text.size=
          document.escape.text.value.length;
      }
    </SCRIPT>
  </HEAD>
```

```
<BODY>
  <FONT size=5 face="Monotype Corsiva">
    <FORM name=escape>
      Le code escape du
      <INPUT type=text size=20 name=text
                    onchange="modify()">est
      <INPUT type=text size=1 name=cod
                                  readonly>.
    </FORM>
  </FONT>
</BODY>
</HTML>
```

Efectul obținut poate fi observat în imaginea următoare:



Funcția eval()

Cu ajutorul funcției `eval()` poate fi realizată evaluarea unui șir de caractere care este considerat a fi o secvență de instrucțiuni *JavaScript*; sintaxa acestei funcții este `eval(șir)`. Vă prezentăm în continuare câteva exemple de utilizare a acestei funcții:

• În urma executării unei secvențe de forma:

```
var x=5;
var y=3;
z=eval("x+y+y+y");
```

variabila `z` va avea valoarea `14`.

• Evaluarea unei variabile va returna valoarea variabilei respective; în urma executării secvenței

```
var x=2457;
z=eval("x");
```

variabila `z` va avea valoarea `2457`.

• Pot fi folosite și funcții sau metode *JavaScript*; în urma executării secvenței

```
z = eval("Math.round(3.14)");
```

variabila `z` va avea valoarea `3`.

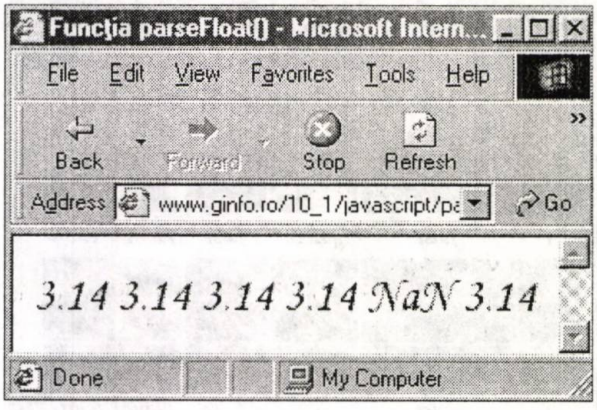
Funcția parseFloat()

Deseori șirurile de caractere conțin valori numerice; uneori este util să putem accesa această valoare ca și cum ar fi un număr și nu un șir de caractere. Pentru a realiza conversia de la șirurile de caractere la valori numerice reale a fost introdusă funcția `parseFloat()` care are ca parametru un șir de caractere și returnează valoarea corespunzătoare șirului respectiv.

În cazul în care este întâlnit un caracter care nu poate face parte dintr-un număr, restul caracterelor din șir sunt ignorate. Următorul exemplu ilustrează modul de utilizare a funcției `parseFloat()`:

```
<HTML>
  <HEAD>
    <TITLE>
      Funcția parseFloat()
    </TITLE>
    <SCRIPT language="JavaScript">
      var x="3.14";
      var y="abc3.14def";
      var z="3.14cd";
      document.write("<FONT face=");
      document.write("'Monotype Corsiva'");
      document.write("size=5>");
      document.write(parseFloat("3.14")+" ");
      document.write(parseFloat("314e-2")+
                                          " ");
      document.write(parseFloat(".0314e+2")+
                                          " ");
      document.write(parseFloat(x)+" ");
      document.write(parseFloat(y)+" ");
      document.write(parseFloat(z));
    </SCRIPT>
  </HEAD>
</HTML>
```

În urma încărcării documentului HTML anterior fereastra programului de navigare ar putea arăta astfel:



Funcția parseInt()

Pentru realizarea conversiei de la șiruri de caractere la valori numerice întregi a fost introdusă funcția `parseInt()`, care are ca parametri un șir de caractere și un număr întreg `b`; ea returnează numărul întreg corespunzător șirului respectiv scris în baza `b`. Parametrul `b` este opțional, în cazul în care acesta lipsește se consideră că baza are valoarea implicit: `10`. În cazul în care este întâlnit un caracter care nu poate face parte dintr-un număr întreg, restul caracterelor din șir sunt ignorate. Următorul exemplu ilustrează modul de utilizare a funcției `parseInt()`:

```
<HTML>
<HEAD>
  <TITLE>Funcția parseInt()</TITLE>
```

```
  <SCRIPT language="JavaScript">
    document.write("<FONT face='Monotype
                    Corsiva' size=5>");
    document.write(parseInt("C",16)+" ");
    document.write(parseInt("14", 8)+" ");
    document.write(parseInt("12", 10)+" ");
    document.write(parseInt("12.99")+" ");
    document.write(parseInt("CPA", 16)+" ");
    document.write(parseInt("1100", 2)+" ");
    document.write(parseInt("12*7", 10));
  </SCRIPT>
</HEAD>
</HTML>
```

În urma încărcării documentului HTML anterior fereastra programului de navigare ar putea arăta astfel:

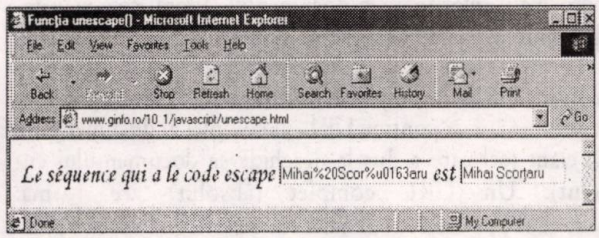
Funcția unescape()

Funcția `unescape()` este folosită pentru a realiza operația inversă funcției `escape()`; ea are ca parametru un șir care reprezintă codificarea în cod escape a unui șir de caractere, decodifică șirul și returnează șirul decodificat. Următorul document HTML ilustrează modul de utilizare a funcției `escape()`.

```
<HTML>
<HEAD><TITLE> Funcția unescape() </TITLE>
  <SCRIPT language=JavaScript>
    function modify(){
      document.unescape.cod.value=unescape
            (document.unescape.text.value);
      document.unescape.cod.size=
          document.unescape.cod.value.length;
      document.unescape.text.size=
         document.unescape.text.value.length;
      }
    </SCRIPT>
  </HEAD>
```

```
  <BODY>
    <FONT size=5 face="Monotype Corsiva">
      <FORM name=unescape>
        Le séquence qui a le code escape
        <INPUT type=text size=20 name=text
                    onchange="modify()">est
        <INPUT type=text size=1 name=cod
                                readonly>.
      </FORM>
    </FONT>
  </BODY>
</HTML>
```

Efectul obținut poate fi observat în imaginea următoare:



Proprietăți specifice tuturor obiectelor

Există două proprietăți care se numără printre proprietățile oricărui obiect *JavaScript* și anume `name` și `length`.

Proprietatea name

Proprietatea `name` este folosită pentru a identifica un anumit obiect, iar valoarea sa poate fi stabilită prin intermediul atributului **name** al marcajului HTML cu ajutorul căruia obiectul respectiv a fost inserat în pagină.

Această proprietate mai are un rol (mult mai important), ea desemnând și identificatorul care poate fi folosit pentru a accesa variabila respectivă. De exemplu, în cazul secvenței:

```
<FORM name=formular>
  <INPUT type=text name=denumire>
</FORM>
```

`denumire` nu va fi numai valoarea proprietății **name**, ci va fi și identificatorul variabilei corespunzătoare casetei de tip text.

Proprietatea length

Proprietatea `length` a unui obiect indică, de obicei, care este dimensiunea obiectului respectiv sau exprimă câte date conține acesta.

Semnificația acestei proprietăți variază în funcție de tipul obiectului; de exemplu, pentru șiruri de caractere ea are ca valoare numărul caracterelor din șir, în cazul vectorilor indică numărul elementelor vectorului etc.

În cazul obiectelor predefinite `length` este o proprietate protejată la scriere (*readonly*), deci valoarea ei nu

poate fi modificată în mod direct, ci doar folosită. Pentru șirurile definite de utilizator proprietatea poate fi folosită pentru a indica dimensiunea șirurilor respective; o secvență de forma: `MaSuite.length=10` definește `MaSuite` ca fiind un șir care conține zece elemente.

Proprietățile obiectului document

Obiectul document înglobează toate elementele conținute într-o pagină Web; acest obiect conține numeroase alte obiecte cum ar fi `anchor`, `form`, `history`, `link`, dar are și câteva proprietăți care îi sunt caracteristice.

Proprietăți informaționale

Există câteva proprietăți cu ajutorul cărora pot fi accesate informații referitoare la documentul curent; cele mai importante astfel de proprietăți sunt `location`, `refferer`, `title`, `lastModified` și `cookie`. Proprietatea `location` conține URL-ul complet al documentului curent; proprietatea `refferer` conține URL-ul complet al documentului apelant (cel care a dus la deschiderea documentului curent). Un URL complet (absolut) are forma: `protocol://domeniu/cale/document`.

Proprietatea `title` conține titlul documentului și este echivalentă cu marcajul HTML **<TITLE>**; Proprietatea `lastModified` conține data și ora ultimei modificări; proprietatea `cookie` conține cookie-ul documentului curent (informații speciale specifice fiecărui utilizator).

Proprietăți pentru afișare

Există cinci proprietăți ale obiectului `document` care corespund culorii fundalului, textului, link-urilor nevizitate, link-urilor active și link-urilor nevizitate.

Proprietatea `bgColor` corespunde culorii de fundal și este echivalentă cu atributul **bgcolor** al marcajului **<BODY>**.

Proprietatea `fgColor` corespunde culorii textului și este echivalentă cu atributul **text** al marcajului **<BODY>**.

Proprietatea `linkColor` corespunde culorii legăturilor nevizitate și este echivalentă cu atributul **link** al marcajului **<BODY>**.

Proprietatea `alinkColor` corespunde culorii legă-turilor active și este echivalentă cu atributul **alink** al marcajului **<BODY>**.

Proprietatea `vlinkColor` corespunde culorii legăturilor vizitate și este echivalentă cu atributul **vlink** al marcajului **<BODY>**.

Valorile acestor proprietăți pot fi șiruri de caractere care desemnează anumite culori sau codificarea RGB a unei culori.

Valorile proprietăților `linkColor`, `alinkColor` și `vlinkColor` pot fi modificate numai înaintea terminării formatării documentului (înainte să înceapă afișarea paginii Web corespunzătoare). Valorile proprietăților `bgColor` și `fgColor` pot fi modificate în orice moment.

Proprietăți pentru conținut

Alături de textul afișat, documentul conține legături, ancore și formulare; în cadrul limbajului *JavaScript* acest conținut suplimentar este grupat în trei proprietăți ale obiectului `document` și anume șirurile `anchors`, `links` și `forms`.

Proprietățile formularelor

Anterior am amintit câteva dintre proprietățile câmpurilor din formulare; formularele au și ele câteva proprietăți specifice alături de proprietățile `name` și `length`, comune tuturor obiectelor *JavaScript*.

Patru dintre proprietățile formularelor sunt echivalente cu atributele marcajului **<FORM>** cu ajutorul căruia a fost inserat formularul. Aceste proprietăți sunt `encoding`, `method`, `action` și `target` și corespund atributelor **enctype**, **method**, **action** și **target**.

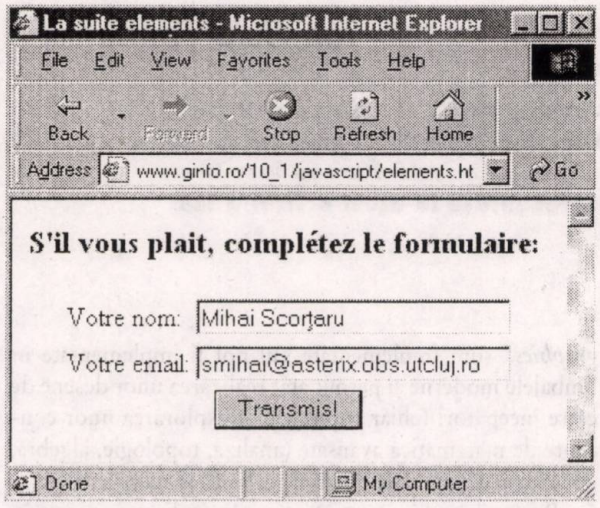
A cincea proprietate a formularelor este un vector (`elements`) care conține obiectele corespunzătoare câmpurilor din formular. Ordinea în care apar câmpurile în cadrul vectorului `elements` este ordinea în care apar marcajele corespunzătoare acestora.

Următorul exemplu ilustrează modul de utilizare al vectorului `elements`; va fi creat un formular și, în momentul apăsării butonului ***Transmis!***, va apărea o casetă de confirmare în interiorul căreia vor apărea informațiile completate.

```
<HTML>
  <HEAD>
    <TITLE>
      La suite elements
    </TITLE>
    <SCRIPT language="JavaScript">
      function verifica(formular) {
        strOut="";
        for(i=0;i<formular.elements.length;
                                    i++)
          strOut+=formular.elements[i].name+
            ":"+formular.elements[i].value+
                                    "\n";
        strOut+="\nEst-ce que vous
                confirmez la transmision?";
        return confirm(strOut);
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <H3>S'il vous plait, complétez le
                              formulaire:</H3>
    <FORM method=post
          onsubmit="return verifica(this)">
      <TABLE>
        <TR><TD>Votre nom:
            <TD><INPUT type=text name=Nume
                                    size=25>
```

```
        <TR><TD>Votre email:
            <TD><INPUT type=text name=Email
                                    size=25>
        <TR><TD align=center colspan=2>
              <INPUT type=submit value=
                "Transmis!" name=Transmite>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

După încărcarea documentului anterior fereastra programului de navigare ar putea arăta astfel:



După apăsarea butonului ***Transmis!*** va apărea următoarea casetă de validare:

Informaţiile vor fi transmise numai în cazul apăsării butonului ***Ok***. Putem accesa toate elementele vectorului `elements` şi folosind instrucţiunea `for...in`; în acest caz funcţia `verifica()` arată astfel:

```
function verifica(formular) {
  strOut="";
  for obj in formular
    strOut+=obj.name+":"+obj.value+"\n";
  strOut+="\nEst-ce que vous confirmez la
                                transmision?";
  return confirm(strOut);
}
```

Proprietatea checked

Proprietatea `ckecked` indică valoarea logică ce corespunde unei casete de validare sau unui buton radio. Starea casetei sau a butonului poate fi schimbată prin intermediul instrucţiunilor *JavaScript* şi schimbările vor fi vizibile imediat pe ecran.

Proprietatea defaultChecked

Proprietatea `defaultChecked` permite alegerea opţiunii selectate implicit în cadrul unui grup de butoane radio.

Proprietatea defaultStatus

Bara de stare conţine, de obicei, informaţii referitoare la încărcarea documentului sau la legătura hipertext deasupra căreia se află cursorul mouse-ului. După încărcarea documentului mesajul implicit afişat în bara de stare este *Document: Done*. Pentru a modifica valoarea implicită poate fi folosită proprietatea `defaultStatus` a obiectului `document`.

Proprietatea href

Proprietatea `href` poate fi folosită pentru a seta URL-ul locaţiei de la care va fi încărcată pagina Web după efectuarea unui click în zona corespunzătoare legăturii hipertext respective.

Mihai Scorţaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro

news...

Intel Celeron la 533 MHz

Primul procesor al anului 2000 este un Celeron. Acest nou procesor de la Intel are o frecvenţă de 533 MHz, magistrală la 66 MHz, un cache level 2 de 128 KB şi este fabricat folosind tehnologia de 0.25 microni. Se pare că acesta este ultimul procesor de la Intel care foloseşte această tehnologie, primul Celeron care va fi fabricat folosind tehnologia de 0.18 microni va apărea în cursul primului trimestrul al anului 2000, va avea o frecvenţă de 566 MHz şi va avea tot o magistrală de 66 MHz. Intel nu va trece încă la magistrale de 100 MHz pentru procesoarele din această familie pentru a nu creşte prea mult preţurile.

Grafica TURTLE și aplicații în PASCAL

Ion Gelu Tomșa

Se spune că prima broscuță țestoasă putea fi deplasată efectiv pe o coală mare de hârtie și, atunci când era apăsată de copilașul de la grădiniță, lăsa o urmă colorată, iar când nu era apăsată, putea fi plimbată fără să lase urme. De la această jucărie și până la aplicațiile prezentate în acest articol a fost însă o cale lungă.

Ce este LOGO?

"Geometria turtle" sau *"grafica turtle"* reprezintă un stil de grafică pe calculator introdus de limbajul de inteligență artificială LOGO. Limbajul LOGO a fost creat de către *Seymour Papert* și colaboratorii săi de la Massachusetts Institute of Technology (*MIT*), care au pornit de la teoria învățării și de la ideile lui *Piaget* privind reorientarea educației pe baza psihologiei copilului. Sigur, se pune întrebarea privitoare la motivul care a stat la baza creării unui nou limbaj de programare în momentul în care existau deja, la dispoziția utilizatorilor/programatorilor o multitudine de limbaje de programare. Răspunsul este legat de faptul că fiecare limbaj de programare, prin facilitățile și lipsurile sale, favorizează formarea și utilizarea unui stil propriu de a realiza programe, dar nici unul nu reprezintă o bază optimă care să stimuleze la maxim calitățile care sălășluiesc în stare latentă la copii, cum sunt: capacitatea de modelare și rezolvare de probleme și spiritul de explorare. În plus, unele din limbajele de programare alese pentru inițiere în informatică (datorită comodității lor în utilizare) conduc către un stil de programare în disonanță cu cele moderne, clare și eficiente. Limbajul LOGO, în schimb, are încorporate toate conceptele moderne care s-au impus în ultimii ani în informatică. Astfel, LOGO permite, pe lângă uzualele calcule aritmetice, și manipularea cu ușurință și naturalețe a cuvintelor și frazelor, fiind astfel adaptat explorării limbajelor naturale și artificiale. Aceasta se datorează în parte limbajului de inteligență artificială LISP (List Procesing) de la care creatorii limbajului LOGO au preluat modul de prelucrare a listelor. În plus, o mare parte a succesului limbajului LOGO rezidă în facilitățile sale grafice. Aceste facilități, cunoscute sub numele de *"turtle graphics"* sunt implementate sau pot fi implementate în limbajele moderne și permit atât realizarea unor desene de către începători (chiar copii), cât și explorarea unor concepte de matematică avansate (analiză, topologie, algebră, geometrie diferențială, mecanică clasică și relativistă etc.).

Prin succesul înregistrat, prin răspândirea sa din ce în ce mai mare, prin revoluționarea sistemelor de instruire, LOGO a depășit granițele unui limbaj obișnuit de programare, devenind un nou stil, o nouă metodă de gândire și învățare.

Grafica turtle în LOGO

Grafica turtle reprezintă un set simplu și puternic de comenzi pentru dirijarea pe ecran a unei "broscuțe țestoase" (turtle). Această denumire a rămas din cauză că prima versiune de LOGO era folosită pentru a comanda de la distanță un robot asemănător cu o broscuță țestoasă. În cazul graficii LOGO broscuța este un pointer pe ecranul grafic care indică locul unde se află aceasta. Forma pointerului este aceea a unui mic triunghi isoscel la care înălțimea dusă prin vârful format de laturile egale arată orientarea (direcția) "broscuței" (cursorului grafic). O explicație foarte ușor de înțeles, chiar pentru începătorii în programarea calculatoarelor, a modului în care desenează broscuța este următoarea: broscuța țestoasă poartă în spate un rucsac în care are creioane (de diferite culori și grosimi) cu care poate desena în timpul deplasării pe ecranul asimilat cu o foaie de hârtie sau un plan în care se deplasează broscuța. Astfel, cu ajutorul unor comenzi simple, ușor de înțeles și folosit, transmitem "broscuței" (cursorului grafic) traseul pe care să-l urmeze și când să deseneze. Deplasările în plan ale broscuței sunt mișcări de tipul *deplasare înainte*, *înapoi*

pe direcția curentă, schimbări ale direcției prin rotiri în plan care corespund geometriei analitice în plan. Cu acest set simplu de comenzi, la care se adaugă altele la fel de simple, se obține un mediu de grafică foarte puternic cu care pot fi modelate toate aspectele geometriei plane. De asemenea, există o extensie a acestui model în spațiu care este folosit efectiv la desenarea și modelarea suprafețelor 3D, cu aplicații în proiectarea pe calculator în domenii din cele mai avansate.

Ecranul grafic LOGO are originea axelor de coordonate în centrul ecranului grafic și orientarea axelor ca în figura 1. Principalele comenzi LOGO pentru deplasarea relativă a pointerului grafic în plan sunt următoarele:

1) `Home` - *mută broscuța în poziția și direcția inițială*: centrul ecranului și direcția spre `Nord` (figura 1). Vârful triunghiului isoscel coincide cu pointerul modului grafic.

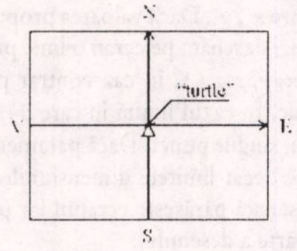


Figura 1

2) `Forward d (FD d)` - *deplasează broscuța înainte pe direcția curentă pe un drum de lungime* `d`. În figura 2, deplasarea se face din punctul inițial P_1 în punctul P_2.

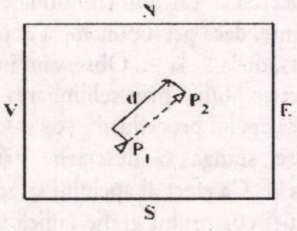


Figura 2

3) `Back d (BK d)` - *deplasează broscuța înapoi pe direcția curentă pe un drum de lungime* `d`. În figura 2, deplasarea se face din punctul inițial P_2 în punctul P_1.

4) `Left u (LT u)` - *schimbă direcția broscuței cu* `u` *grade spre stânga* (în sens opus deplasării acelor de ceasornic) față de direcția curentă. În figura 3, din punctul inițial P_1 pointerul grafic ajunge în punctul P_2.

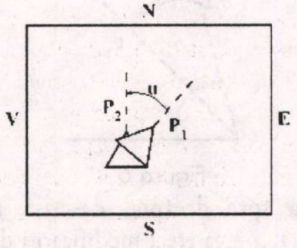


Figura 3

5) `Right u (RT u)` - *schimbă direcția broscuței cu* `u` *grade spre dreapta* (în sensul acelor de ceasornic) față de direcția curentă. În figura 3, din punctul inițial P_2 pointerul grafic ajunge în punctul P_1.

6) `Creiondown (PD)` - *creion jos*: deplasarea broscuței lasă urme pe ecran.

7) `CreionUp (PU)` - *ridică creionul*: broscuța se deplasează fără să lase urme pe ecran.

8) `SetPos nx,ny` - *mută* broscuța (pointerul grafic) în punctul de coordonate absolute `P(nx,ny)`.

9) `SetHeading u (SETH u)` - *schimbă direcția țestoasei* la direcția dată de unghiul `u` față de orientarea spre `Nord`. Argumentul `u` reprezintă orientarea în grade față de sensul pozitiv al axei `Oy` (`Nord`).

10) `ShowTurtle (ST)` - *face vizibilă pe ecran poziția cursorului grafic* (triunghi isoscel) în poziția curentă a țestoasei.

11) `HideTurtle (HT)` - *ascunde cursorul grafic de pe ecran*, ceea ce duce la mărirea vitezei de desenare.

Observații:

i) Parametrii comenzilor de deplasare și rotire sunt numere întregi, interpretate astfel:
- la deplasarea înainte (`FORWARD`) direcția deplasării este dată de înălțimea triunghiului isoscel (reprezentarea *"turtle"*), iar sensul deplasării este indicat de pointerul grafic, pentru valori pozitive ale distanței (`d>0`), sau sens opus pentru valori negative ale distanței (`d<0`). În cazul deplasării înapoi sensul se inversează.
- la fel, în cazul schimbării direcției cu comanda `LEFT u`, valorile pozitive ale unghiului `u` sunt interpretate ca o rotire spre stânga (sens opus acelor de ceasornic), iar cele negative ca o rotire spre dreapta. Pentru comanda `RIGHT u` sensul rotirii se inversează.

ii) Denumirea unor comenzi `LOGO` poate fi redusă la câteva caractere (două de cele mai multe ori) care au fost scrise între paranteze acolo unde este cazul, de exemplu `FORWARD=FD` ș.a.m.d.

Grafica turtle în PASCAL

Comenzile graficii turtle pot fi implementate în orice alt limbaj de programare care permite folosirea ecranului în mod grafic și apelarea unor primitive grafice de desenare a punctelor și liniilor.

Limbajul PASCAL oferă prin interfața unit-ului `Graph` procedurile pentru implementarea acestor comenzi.

Coordonatele pointerului grafic le memorăm în două variabile `x`, `y` de tip `Real`, corespunzătoare unui ecran virtual cu originea `O(0,0)` în centrul ecranului fizic (vezi figura 4).

Coordonatele se modifică odată cu deplasarea pe ecran a pointerului grafic. Orientarea "țestoasei" este dată de unghiul `t` pe care îl face semidreapta ce unește originea axelor și punctul `P(x,y)` cu axa `Ox`. În loc să memorăm va-

loarea unghiului t, vom păstra în memorie valorile funcțiilor sin și cos în variabilele sint, cost.

Pentru a preciza dacă în timpul deplasării se desenează traiectoria, folosim valoarea booleană creion. Dacă creion este true înainte de apelarea unei comenzi de deplasare, pe ecran va apare desenată traiectoria, în caz contrar nu se desenează nimic. Culoarea liniei desenate se stabilește cu procedura standard SetColor (din unit-ul Graph).

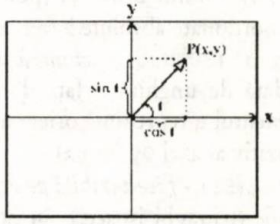


Figura 4

Declarăm următoarele proprietăți pentru cursorul grafic (*turtle*):

```
var
  culoare:Integer;
  creion:Boolean;
  cost,sint:Real;
  x,y:Integer;
```

Comenzile necesare pentru dirijarea pointerului grafic le implementăm sub forma unor proceduri PASCAL descrise în unit-ul **TURTLE.PAS**. Pentru a putea descrie procedurile pentru deplasarea și rotirea cursorului grafic trebuie să deducem formulele analitice ale coordonatelor punctelor destinație.

1) Procedura HOME inițializează proprietățile cursorului și afișează cursorul grafic în centrul ecranului. Proprietățile sunt inițializate astfel:
   - coordonatele pointerului grafic vor fi
     x=GetMaxX **div** 2, y=GetMaxY **div** 2 ;
   - orientarea sint=1, cost=0 corespunzătoare unghiului t=90° (dat de orientarea spre Nord);
   - culoare:=Black;
   - creion:=true.
2) Deplasarea înainte o descriem prin procedura _FORWARD(d:Real) - scriem prefixul "_" în numele procedurii pentru a nu se confunda cu directiva de compilare PASCAL **forward**. Coordonatele punctului destinație le deducem din triunghiul dreptunghic ΔPP_2P_2, unde P_1 este punctul de plecare, P_2 punctul destinație și P proiecția lui P_2 pe axa Ox (figura 5).

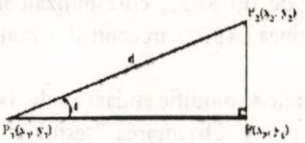


Figura 5

Din figura 5 rezultă că laturile P_1P și PP_2 (catetele tringhiului dreptunghic) sunt:

$$(1)\ \begin{cases} P_1P = d\cdot\cos(t) = x_2-x_1 \\ PP_2 = d\cdot\sin(t) = y_2-y_1 \end{cases}$$

De aici obținem coordonatele punctului P_2:

$$(2)\ \begin{cases} x_2 = x_1+d\cdot\cos(t) \\ y_2 = y_1+d\cdot\sin(t) \end{cases}$$

Ținând cont de proprietățile cursorului grafic, noile coordonate rezultate în urma deplasării pe distanța d se calculează astfel:

$$(3)\ \begin{cases} x = x+d\cdot cost \\ y = y+d\cdot sint \end{cases}$$

În procedura _FORWARD calculăm coordonatele cursorului grafic cu formulele (3) și le păstrăm în variabilele locale intermediare x_1, y_1. Dacă valoarea proprietății creion este true, atunci desenăm pe ecran o linie prin apelul procedurii Line(x,y,x_1,y_1), în caz contrar pe ecran nu se desenează nimic. În cazul limită în care d=1 pe ecran trebuie desenat un singur punct. Dacă parametrul d are o valoare mai mare decât limitele dimensiunilor ecranului în pixeli, linia desenată părăsește ecranul iar pe ecran va rămâne doar o parte a desenului.

Prin apelul acestei proceduri nu se modifică decât coordonatele cursorului grafic, celelalte proprietăți rămânând neschimbate.

3) Deplasarea înapoi este descrisă prin procedura BACK(d:Real). Calculul coordonatelor punctului destinație este același cu calculul coordonatelor pentru deplasarea înainte, dacă presupunem că deplasarea se face în sens invers, de la P_2 la P_1. Observăm în formulele (3) că acest efect se obține prin schimbarea semnului distanței d, adică apelul procedurii _FORWARD(-d).
4) Rotirea spre stânga o descriem prin procedura LEFT(u:Real). Ca efect al apelului se actualizează valoarea direcției cursorului grafic (adică unghiul t) corespunzător noii direcții, unghiul t+u (vezi figura 6). Proprietățile cursorului se modifică astfel:

$$(4)\ \begin{cases} \sin(t+u) = \sin(t)\cdot\cos(u) + \cos(t)\cdot\sin(u) \\ \cos(t+u) = \cos(t)\cdot\cos(u) - \sin(t)\cdot\sin(u) \end{cases}$$

$$(5)\ \begin{cases} sint = sint\cdot\cos(u) + cost\cdot\sin(u) \\ cost = cost\cdot\cos(u) - sint\cdot\sin(u) \end{cases}$$

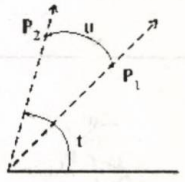


Figura 6

5) Rotirea spre dreapta, descrisă în procedura RIGHT(u:Real), are ca efect modificarea direcției cursorului grafic dată de unghiul t, corespunzător noii direcții,

unghiul t-u (vezi figura 7). Proprietățile cursorului se modifică astfel:

$$(6)\quad \begin{cases} \sin(t-u)=\sin(u)\cdot\cos(t)-\cos(u)\cdot\sin(t) \\ \qquad\quad =\sin(t+(-u)) \\ \cos(t-u)=\cos(u)\cdot\cos(t)+\sin(u)\cdot\sin(t) \\ \qquad\quad =\cos(t+(-u)) \end{cases}$$

Din formulele (6) se observă că rotirea spre dreapta este echivalentă cu rotirea spre stânga cu unghiul -u. Apelul acestei proceduri este echivalent cu apelul LEFT(-u).

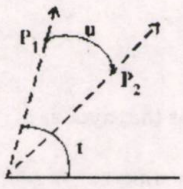


Figura 7

Celelalte proceduri sunt simple actualizări ale unor proprietăți ale cursorului grafic și sunt explicate în cadrul unit-ului prin comentariile inserate în text:

6) Procedura Pen_up - ridică creionul: broscuța se deplasează fără să lase urme pe ecran.

7) Procedura Pen_down - creion jos: deplasarea broscuței lasă urme pe ecran.

8) Procedura SetPos(nx, ny:Real) - mută broscuța în punctul (nx, ny).

9) Procedura Heading(u:Real) - schimbă direcția țestoasei la direcția dată de unghiul u prin valorile sin(u) și cos(u).

10) Procedura Clear - șterge ecranul grafic.

11) Procedura Show_turtle - desenează un cursor sub forma unui mic triunghi isoscel în poziția curentă a broscuței.

12) Procedura Hide_turtle - ascunde (șterge) cursorul grafic. Această procedură este apelată în timpul desenării pentru a mări viteza de desenare, în special în cazul figurilor recursive.

Procedurile de mai sus sunt descrise în unit-ul **TURTLE.PAS** prezentat în continuare:

Listing unit-ul TURTLE.PAS

```
Unit turtle;
interface
uses Graph;
const _pi=3.14159265;
var
  creion:Boolean;
  cost,sint:Real;
  x,y:Integer;

procedure Start;
{ Initializeaza ecranul grafic si pozitia }
{ broscutei pe ecran in mijlocul ecranului }
{ si cu directia spre Nord.               }
procedure Stop;
            { Iesirea din modul grafic si }
            {     revenirea in modul text. }

procedure Home;
 { Muta broscuta in pozitia si dir. initiala }
 { (centrul ecranului si dir. spre Nord)     }

procedure Creion_up;
 { Ridica creionul => broscuta se deplaseaza }
 {             fara sa lase urme pe ecran.   }

procedure Creion_down;
     { Creion jos => deplasarea broscutei }
     {                lasa urme pe ecran. }

procedure Right(u:Real);
{ Schimba directia broscutei cu u grade spre }
{       dreapta fata de directia curenta.    }

procedure Left(u:Real);
{  Schimba directia broscutei cu u grade spre }
{        stanga fata de directia curenta.     }

procedure _Forward(d:Real);
{ Deplaseaza broscutei inainte pe directia }
{       curenta pe un drum de lungime d.   }

procedure Back(d:Real);
{  Deplaseaza broscuta inapoi pe directia }
{       curenta pe un drum de lungime d.  }

procedure SetPos(nx,ny:Real);
{ Muta broscuta in punctul (nx,ny).       }

procedure Heading(u:Real);
{  Schimba directia testoasei la directia }
{                    data de unghiul u.   }

procedure Clear;
                { Sterge ecranul grafic. }

procedure Show_turtle;
 { Afiseaza un pointer in pozitia curenta }
 {                          a testoasei.  }

procedure Hide_turtle;
                    { Ascunde pointerul. }

implementation

procedure Start;
var gd,gm,err:Integer;
begin
  gd:=Detect;
```

```
  InitGraph(gd,gm,'');
  err:=GraphResult;
  if Err=grOk
  then
  begin                          { Do graphics }
    Setbkcolor(blue);
    Clearviewport;
    x:=GetMaxX div 2;
    y:=GetMaxY div 2;
    MoveTo(x,y);
    cost:=1;
    sint:=0;
    creion:=true
  end
  else
  Writeln('Graphics error:',
                          GraphErrorMsg(Err))
end;

procedure Stop;
begin
  Closegraph
end;

procedure Home;
begin
  x:=GetMaxX div 2; y:=GetMaxY div 2;
  cost:=1; sint:=0; creion:=true;
  MoveTo(x,y)
end;

procedure Creion_up;
begin
  creion:=false
end;

procedure Creion_down;
begin
  creion:=true
end;

procedure Right(u:Real);
var sinu,cosu,st,ct:Real;
begin
  sinu:=Sin(u*_pi/180);
  cosu:=Cos(u*_pi/180);
  st:=sint*cosu-cost*sinu;
  ct:=cost*cosu+sint*sinu;
  sint:=st; cost:=ct
end;

procedure Left(u:Real);
begin
  Right(-u)
end;
```

```
procedure _Forward(d:Real);
var dx,dy:Integer;
begin
  dx:=Trunc(d*cost);
  dy:=-Trunc(d*sint);
  if creion then LineTo(x+dx,y+dy);
  x:=x+dx; y:=y+dy
end;

procedure Back(d:Real);
begin
  _Forward(-d)
end;

procedure SetPos(nx,ny:Real);
begin
  x:=Round(Trunc(nx));
  y:=Round(Trunc(ny));
  MoveTo(x,y)
end;

procedure Heading(u:Real);
begin
  cost:=Cos(u*_pi/180);
  sint:=Sin(u*_pi/180)
end;

procedure Clear;
begin
  ClearViewport
end;

procedure Show_turtle;
begin
  _Forward(5); Creion_up; Back(5);
  Creion_down; Right(150); _Forward(7);
  Right(120); _Forward(7); Right(120);
  _Forward(7); Left(30)
end;

procedure Hide_turtle;
begin
  SetColor(GetBkColor);
  Show_turtle;
  SetColor(White)
end
End.
```

Aplicații la desenarea fractalilor

Cu ajutorul procedurilor graficii turtle putem realiza desene prin programe PASCAL care includ unitatea **TURTLE.PAS** în secțiunea **uses**. Programele scrise de noi trebuie să descrie desenul prin succesiunea de comenzi (proceduri) care să producă efectul urmărit pe ecran, putând fi folosite cu succes atât la fixarea noțiunilor introduse de

grafica pe calculator cât și la explorarea noțiunilor de geometrie, de exemplu desenarea figurilor geometrice, aplicații ale geometriei analitice în plan, aplicații ale transformărilor geometrice etc.

Cele mai spectaculoase aplicații sunt însă cele care desenează figuri geometrice recursive (*fractali*). Pentru desenarea acestor figuri principiile de bază folosite sunt principiul recursivității și al transparenței referențiale.

Ca exemplu, privind utilizarea comenzilor graficii *turtle* la desenarea fractalilor, să considerăm un algoritm pentru desenarea unui arbore binar stilizat. Presupunem că arborele are mai multe stadii de dezvoltare, pe care le notăm în ordine 0, 1, 2, ... Un arbore de vârstă 0 este reprezentat de un punct, un arbore de vârstă 1 este reprezentat de un segment de dreaptă de lungime 1, și în general un arbore de vârstă n este reprezentat de un segment care se ramifică la vârf în două ramuri care, abstracție făcând de orientare, sunt ambele izomorfe cu un arbore de vârsta n-1. Pentru vârstele 0, 1, 2 obținem următoarele modele:

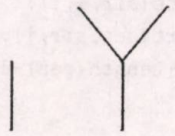


Figura 8

Utilizând primitivele de grafică turtle, algoritmul recursiv care generează un astfel de arbore se obține direct din enunțul problemei și poate fi scris sub forma procedurii:

Listing procedura Arbore

```
procedure Arbore( n:Integer);
begin
  if n=0 then                { deseneaza_punct }
  else
  begin
    _Forward(l);
    Left(45); Arbore(n-1);
    Right(90); Arbore(n-1);
    Left(45); Back(l)
  end
end;
```

Reușita acestui algoritm, ca și a multor altor algoritmi recursivi de grafică *turtle*, se bazează pe respectarea principiului cunoscut sub numele de *"transparență referențială"*. În esență, acest principiu cere ca, pentru îmbinarea corectă a componentelor unui desen, la terminarea fiecărei proceduri cursorul (creionul) să se găsească exact în aceeași poziție (coordonate și orientare) ca la începutul apelării procedurii. Prin inducție, vom verifica pentru algoritmul de mai sus, principiul transparenței referențiale.

Pasul de verificare: într-adevăr, pentru n=0 ipoteza este evident verificată. Pentru n=1, execuția urmează ramura **else**, deci cursorul se deplasează înainte pe direcția NORD, se rotește spre stânga cu 45°, se apelează Arbore cu parametrul n=0, deci se desenează un punct, se rotește spre dreapta cu 90° și se apelează procedura Arbore cu n=0, deci se desenează un punct peste cel desenat anterior, în sfârșit se rotește spre stânga cu 45° (adică se revine la direcția NORD inițială) și se deplasează înapoi pe distanța pe care a parcurs-o la început, prin urmare cursorul va avea direcția NORD și poziția sa va fi exact în punctul inițial, când a avut loc apelul Arbore.

Pasul de inducție: acum, presupunând că ipoteza este verificată pentru cazul n=k≥2, s-o verificăm pentru cazul n=k+1. Prin apelul Arbore cu n=k+1, execuția urmează ramura **else** și, conform celor arătate anterior pentru n=1, schimbarea totală a orientării este egală cu o întoarcere la stânga cu 45°+0°-90°+0°+45°=0°. Similar, deplasarea totală este egală cu 1+0+0-1=0. Prin urmare, în cazul procedurii Arbore este respectat principiul transparenței referențiale.

Arborele desenat cu această procedură este un arbore ideal, cu ramuri de aceeași lungime și toate înclinate cu 45° față de ramura din care provin. Un astfel de arbore nu este întâlnit în realitate, de aceea vom aduce câteva modificări acestei proceduri pentru a obține o reprezentare mai apropiată de arborii pe care îi regăsim în natură. La fiecare stadiu de dezvoltare al arborelui presupunem că lungimea ultimei ramuri este mai mică decât ramura din care provine și reprezintă un procent p din aceasta. De asemenea, mai presupunem că ramurile au o înclinare diferită. Cu aceste modificări procedura Arbore devine:

Listing procedura Arbore

```
procedure Arbore( n:Integer);
begin
  if n=1 then
  begin _Forward(l); Back(l) end
  else
  begin
    l:=l*p; _Forward(l); Left(u1);
    Arbore(n-1); Right(u1+u2); Arbore(n-1);
    Left(u2); Back(l)
  end
end;
```

Variabilele l, p, u1 și u2 sunt variabile globale și au următoarea semnificație: l este lungimea arborelui după primul stadiu de dezvoltare, p procentul de creștere al unei ramuri, u1 și u2 unghiurile de înclinare a ramurilor, u1 spre stânga și u2 spre dreapta. Pentru un grad mai mare de realism putem modifica aleator unghiurile de creștere a ramurilor în cadrul fiecărei generații.

L-sisteme

Procedurile recursive folosite sunt în general mari consumatoare de memorie, de aceea este de preferat varianta nerecursivă. În cazul fractalilor (definiți recursiv) o variantă iterativă de desenare o reprezintă așa-numitele *L-sisteme*.

În general, pentru desenarea unei figuri recursive (fractal) distingem următoarele elemente:

- *inițiator* - este desenul obținut pentru generația 1;
- *generator* - reprezintă transformările aplicate inițiatorului (sau ultimei generații desenate) pentru a trece la următoarea generație.

Aceste elemente, *inițiator* și *generator*, pot fi descrise formal cu ajutorul *L-sistemelor*. *L-sistemele* sunt metode de generare a imaginilor fractale bazată pe concepte elementare și grafica *turtle*. Acestea au fost introduse în cartea ***"The Algorithmic Beauty of Plants"*** scrisă de *Przemysaw Prusinkiewicz* și *Aristid Lindenmayer*. Construcția *L-sistemelor* se bazează pe o tehnică simplă de înlocuire în cadrul unui șir de caractere care este iterată de mai multe ori, obținându-se noi generații. Această tehnică poate crea fractalii comuni, dar este deseori folosită cu succes pentru a genera arbori, arbuști, flori și o varietate de alte plante.

Simbolurile elementare folosite în cadrul unui *L-sistem* sunt următoarele:

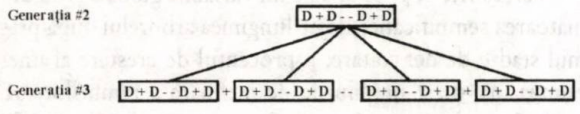
d: mută cursorul (*turtle*) înainte cu o unitate fără a desena o linie;
D: mută cursorul (*turtle*) înainte cu o unitate și desenează o linie;
+: întoarce cursorul (*turtle*) spre dreapta cu un unghi dat;
-: întoarce cursorul (*turtle*) spre stânga cu un unghi dat.

Distanța de deplasare, ca și unghiul de întoarcere sunt precizate înainte de a genera *L-sistemul*.

De exemplu, pentru a desena *curba lui Koch*, folosim inițiatorul D, care desenează o linie pentru generația #1, și *D-generatorul* D+D- -D+D, cu unghiul de întoarcere 60°, cu care desenăm generația #2:

Figura 9

Trecerea la generația următoare în *L-sistemul* asociat se face înlocuind fiecare literă D cu *D-generatorul*. Toate simbolurile pentru care avem generator vor fi înlocuite cu generatorul respectiv în timp ce restul simbolurilor se păstrează neschimbate. Astfel, obținem din:



Un alt exemplu, pentru a desena *fulgul de nea*, avem inițiatorul: D- -D- -D, pentru unghiul de întoarcere 60°, care desenează triunghiul echilateral pentru generația 1. *D-generatorul* este cel folosit la *curba lui Koch*: D+D--D+D.

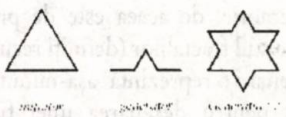
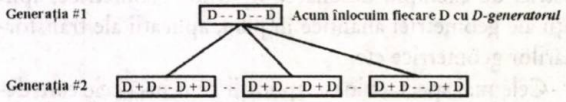


Figura 10

În acest caz, *L-sistemul* se construiește astfel:



Procedura care realizează trecerea la generația următoare prin substituirea simbolului D cu *D-generatorul* este dată în continuare:

Listing procedura Next_generation

```
procedure next_generation(var sir:string);
var i:Integer;
begin
  i:=1;
  while i<=Length(sir) do
  begin
    if sir[i]='D' then
    begin
      Delete(sir,i,1);
      Insert(gen,sir,i);
      i:=i+Length(gen)-1
    end
    Inc(i)
  end
end;
```

În această procedură variabila `gen` păstrează *D-generatorul*, adică `gen:="D+D-D+D"`.

Desenarea *L-sistemului* se face simplu prin parcurgerea secvenței de operații de desenare descrise de șirul care reprezintă *L-sistemul* astfel:

Listing procedura Draw

```
procedure Draw(sir:string);
var i:Integer;
begin
  Start;
  for i:=1 to Length(sir) do
    case sir[i] of
      'D': _Forward(1);
      'd': begin
             PenUp; _Forward(1); PenDown
           end;
      '-': Left(u);
      '+': Right(u)
    end
end;
```

În această formă dimensiunea maximă a unui *L-sistem* construit la o anumită generație este limitată de dimensiunea maximă a tipului **string** în PASCAL, adică 255 caractere. Pentru a depăși această limită și pentru a construi *L-sisteme* foarte mari se pot genera *L-sistemele* în fișiere text, iar desenarea lor se face prin parcurgerea secvențială

a fișierului. Astfel, pot fi create *L-sisteme* virtuale foarte mari, de generații mai înaintate. Acest lucru nu este posibil în cazul procedurilor recursive de desenare, când din cauza apelului recursiv spațiul alocat în stiva de execuție se epuizează.

Lăsăm pe seama cititorului adăugarea unei interfețe grafice (în DOS sau WINDOWS) pentru desenarea generațiilor de fractali descriși recursiv cât și ca *L-sisteme*.

Alți fractali sunt descriși și comentați în textul sursă al programului:

- *arborele lui Perron*;
- *curba lui Sierpinski*: are proprietatea că - la infinit - este o curbă de lungime infinită delimitată de o suprafață de arie finită, și anume de un pătrat;
- *covorul lui Sierpinski*: reprezintă un exemplu de obiect geometric despre care nu s-a putut preciza dacă este o curbă ori o suprafață;
- *curba lui Hilbert*: este un exemplu de curbă continuă de lungime infinită care „umple" un pătrat, adică este o funcție $f:[0,1]\to[0,1]\times[0,1]$ continuă și surjectivă, dar neinjectivă;
- *curba lui Peano*: este un alt exemplu de curbă care are proprietatea curbei lui *Hilbert*;
- *curba Dragonului*: descrisă de *Gardner* și *Kroger*;
- *spirala lui Arhimede*: modelează procesul creșterii cochiliilor de melci și al evoluției în general (*curba "învățării"*);
- *curba C*.

În continuare este prezentat programul sursă complet pentru desenarea fractalilor prezentați prin proceduri recursive.

Listing FRACTALI.PAS

```
Program Fractali;
uses turtle,Graph,Crt;
var m,n:Integer;
    l:Real;

procedure linie(n:Integer;l:Integer);
                                { curba Koch }
begin
  if n=0 then _Forward(l)
  else
  begin
    linie(n-1,l div 3); Left(60);
    linie(n-1,l div 3); Right(120);
    linie(n-1,l div 3); Left(60);
    linie(n-1,l div 3)
  end
end;

procedure fulg(n:Integer;l:Integer);
                    { curba Koch pe triunghi }
begin
  linie(n,l); Right(120); linie(n,l);
  Right(120); linie(n,l); Right(120)
end;

procedure fulg_2(m,n:Integer;l:Integer);
{ curba Koch pe un polig. regulat cu m lat. }
var i:Integer;
begin
  for i:=1 to m do
  begin
    linie(n,l); Right(180-(m-2)*180/m)
  end
end;

procedure arbore_binar(n:Integer;l:Real);
                { n - generatia; l - lungimea }
begin
  if n>0 then
  begin
    _Forward(l); Left(45);
    arbore_binar(n-1,l*0.6); Right(90);
    arbore_binar(n-1,l*0.6); Left(45);
    Back(l)
  end
end;

procedure Perron(n:Integer;l,p,u:Real);
 { n - nr. generatiei; l - lung. de crestere }
 { p - procentul de crestere (recomandat <=1 }
 {                  u - unghiul de creștere }
begin
  if n>0 then
  begin
    Left(u); _Forward(l);
    Perron(n-1,l*p,p,u);
    Back(l); Right(2*u); _Forward(l);
    Right(u); _Forward(l);
    Perron(n-1,l*p,p,u);
    Back(l); Left(2*u); _Forward(l);
    Left(u); _Forward(l);
    Perron(n-1,l*p,p,u);
    Back(l); Right(2*u); _Forward(l);
    Perron(n-1,l*p,p,u);
    Back(l); Left(u); Back(l);
    Right(u); Back(l); Left(u)
  end
end;

procedure curba_Sierpinski(n,l:Integer);
                          { n - generatia }
                        { l - lungimea laturii }
var d:Real;
begin
  if n>0 then
  begin
    d:=l/Sqrt(2); curba_Sierpinski(n-1,l);
```

```
    Right(45); _Forward(d); Right(45);
    curba_Sierpinski(n-1,l); Left(90);
    _Forward(d); Left(90);
    curba_Sierpinski(n-1,l); Right(45);
    _Forward(d); Right(45);
    curba_Sierpinski(n-1,l)
  end
end;

procedure patrat(px,py,l:Real);
          { folosita in procedura Covor }
begin
  SetPos(px,py); Heading(0); _Forward(l);
  Right(90); _Forward(l); Right(90);
  _Forward(l); Right(90); _Forward(l);
  Right(90)
end;

procedure Covor(n:Integer;x,y,l:Real);
var i:Integer;
begin
  if n=0 then Patrat(x+l/3,y+l/3,l/3)
  else
  begin
    Patrat(x+l/3,y+l/3,l/3);
    for i:=0 to 2 do
      Covor(n-1,x,y+i*l/3,l/3);
    Covor(n-1,x+l/3,y+2*l/3,l/3);
    for i:=0 to 2 do
      Covor(n-1,x+2*l/3,y+(2-i)*l/3,l/3);
    Covor(n-1,x+l/3,y,l/3)
  end
end;

procedure Hilbert(n,l,v:Integer);
{n - generatia; l - lungimea; v - orientarea }
begin
  if n>0 then
  begin
    Left(v*90); Hilbert(n-1,l,-v);
    _Forward(l); Right(v*90);
    Hilbert(n-1,l,v); _Forward(l);
    Hilbert(n-1,l,v); Right(v*90);
    _Forward(l); Hilbert(n-1,l,-v);
    Left(v*90)
  end
end;

procedure Peano(n:Integer;l:Integer);
          { n - generatia; l - lungimea }
begin
  if n=0 then _Forward(l)
  else begin
         Peano(n-1,l div 3); Left(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3); Left(90);
         Peano(n-1,l div 3); Left(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3); Left(90);
         Peano(n-1,l div 3); Left(90);
         Peano(n-1,l div 3); Right(90);
         Peano(n-1,l div 3)
       end
end;

procedure Dragon(n,l,v:Integer);
{ n - generatia; l - lungimea; v - orientare }
begin
  if n=0 then _Forward(l)
  else begin
         Right(45*v); Dragon(n-1,l,1);
         Left(90*v); Dragon(n-1,l,-1);
         Right(45*v); Delay(10)
       end
end;

procedure Spirala(n,k:Integer;l,u:Real);
begin
  if n>=k then
  begin
    Left(u); _Forward(l); Delay(100);
    Spirala(n-1,k+1,1.1*l,u)
  end
end;

procedure Koch(n,l:Integer);
          { n - generatia; l - lungimea }
begin
  if n=0 then _Forward(l)
  else begin
         Koch(n-1,l div 4); Left(90);
         Koch(n-1,l div 4); Right(90);
         Koch(n-1,l div 4); Right(90);
         Koch(n-1,l div 4);
         Koch(n-1,l div 4); Left(90);
         Koch(n-1,l div 4); Left(90);
         Koch(n-1,l div 4); Right(90);
         Koch(n-1,l div 4)
       end
end;

procedure curba_C(n,l:Integer);
{ n - generatia; l - lungimea }
begin
  if n=0 then _Forward(l)
  else begin
         curba_C(n-1,l); Right(90);
         curba_C(n-1,l); Left(90)
       end
end;
```

PC MAGAZINE ROMANIA

COMPUTER PRESS AGORA

Din acest număr:

Premiile de excelență tehnică PC Magazine

- Camere digitale
- Romtelecom: de la monopol la om
- Recunoașterea vorbirii
- Infracțiunea informatică

www.pcmagazine.ro

Revista utilizatorilor de calculatoare personale

COMPUTER PRESS AGORA

Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300

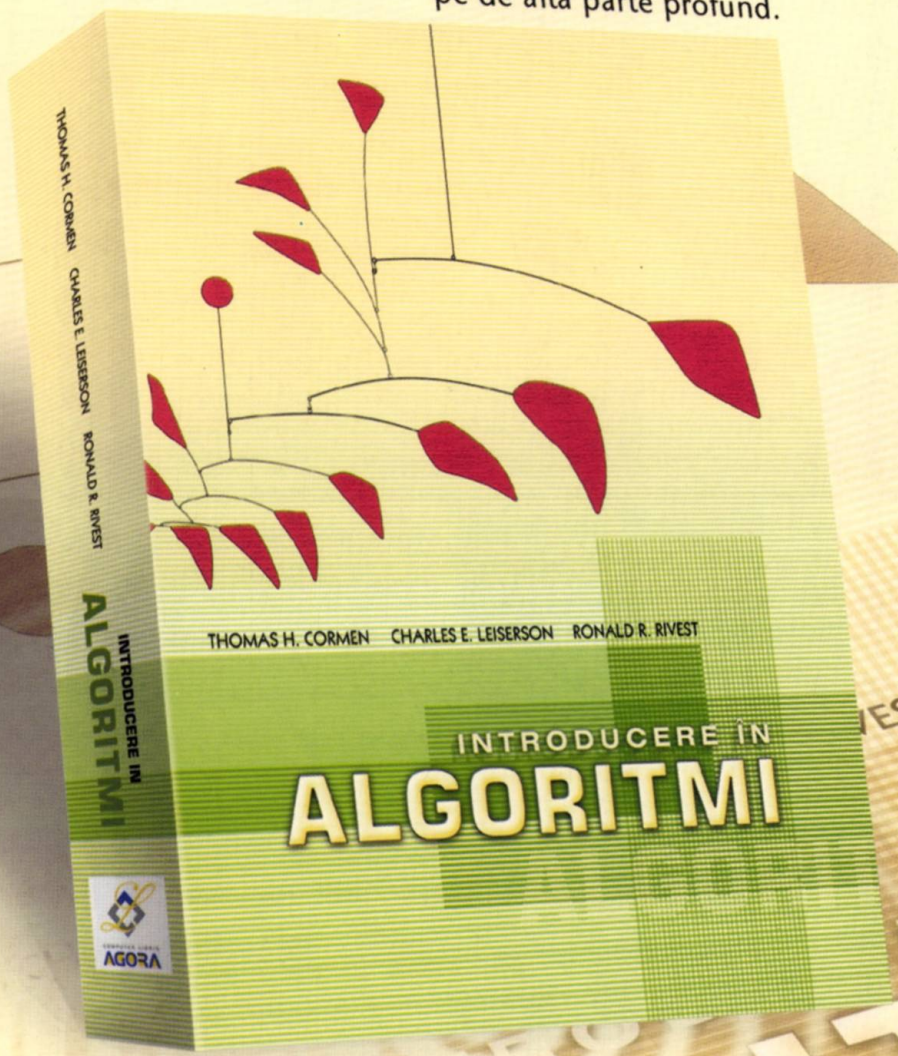
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

```
procedure meniu;
var c:Char;
    n,l:Integer;
    p
begin
  repe
    Wr                urba lui Koch');
    Wr                lg de nea');
    Wri               bore binar');
    Wri               bore Perron');
    Wri               ba lui Sierpinski');
    Writ              orul lui Sierpinski');
    Writ              ba lui Hilbert');
    Writ              ba lui Peano');
    Write             a dragonului');
    Write             ala lui Arhimede');
    Write             a lui Koch - poligon
                                  cu m laturi');
    Writel            C');
    Writel            );
    Write(            s.: ');
    c:=Read
    if c=#0           ;
    case up
      '0': Exit;
      '1': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; linie(n,l)
           end;
      '2': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; fulg(n,l)
           end;
      '3': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; Heading(90);
             Setpos(x,2*y-10);
             arbore_binar(n,l);
           end;
      '4': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Write(' p='); Readln(p);
             Write(' u='); Readln(u);
             Start; Heading(90);
             Setpos(x,2*y-20);
             _Forward(l);
             Perron(n,l,p,u)
           end;
      '5': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; Heading(180);
             curba_Sierpinski(n,l)
           end;
      '6': begin
             Write(' n='); Readln(n);
             Write('px='); Readln(px);
             Write('py='); Readln(py);
             Write(' l='); Readln(l);
             Start; Heading(180);
             covor(n,px,py,l)
           end;
      '7': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; SetPos(x,2*y-20);
             Heading(90); Hilbert(n,l,1)
           end;
      '8': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; Peano(n,l)
           end;
      '9': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; Dragon(n,l,1)
           end;
      'A': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Write(' u='); Readln(u);
             Start; Clear; Home;
             Spirala(n,1,l,u)
           end;
      'B': begin
             Write(' n='); Readln(n);
             Write(' m='); Readln(m);
             Write(' l='); Readln(l);
             Start; fulg_2(m,n,l)
           end;
      'C': begin
             Write(' n='); Readln(n);
             Write(' l='); Readln(l);
             Start; curba_C(n,l)
           end
    end;
    Outtextxy(260,460,'Apasati o tasta!');
    if Readkey=#0 then Readkey;
    Stop
  until c='0'
end;

Begin
  meniu
End.
```

CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS CARTE POȘTALĂ POST RĂSPUNS

Este în curs de apariție și în limba română celebra carte de ALGORITMI scrisă de Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, publicată de MIT Press, SUA. Este o carte care de la prima ei apariție a fost retipărită, numai în Statele Unite, de 21 ori.

Această carte oferă o introducere exhaustivă a studiului modern al algoritmilor. Analiza și proiectarea algoritmilor tratați este accesibilă și asigură condițiile unui studiu, pe de o parte fundamental, pe de altă parte profund.



COMPUTER LIBRIS AGORA

Pentru comenzi sau informații suplimentare puteți suna la tel. 064-192422

Cure-all?

For a great night's sleep, try Compaq Deskpro. It relieves the headaches of day-to-day PC management with powerful features like the new Compaq Insight Manager LC. It's the industry's first integrated tool for managing workgroups of PCs from a simple, browser-based interface. So you spend less time playing doctor and more time getting real work done. Just another healthy dose of exceptional value from Compaq -- and another reason Deskpro is number one. For more therapeutic thoughts, visit www.compaq.com /deskpro, or contact us at phone: +40-1-3033100

COMPAQ

OMNILOGIC BGS

135, Bucureşti-Ploieşti Road,
Bucharest 1, Romania
PO BOX 18-131
Tel: +40-1-303 31 00
Fax: +40-1-303 31 64
www.omnilogic.ro

Intel, the Intel Inside logo and Pentium are registered trademarks of Intel Corporation

Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

Februarie 2000
Vol. 10/2

Lei 15.000

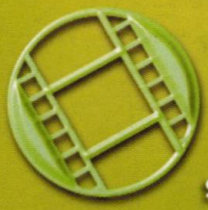
din sumarul numărului:

Programarea dinamică în problemele NP-complete

CONCURSURI

Rezultate Runda 3

Clasament
Enunțuri Runda 5



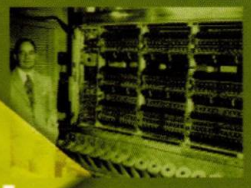
SERIAL

Teoria grafurilor
Pagini Web cu JavaScript

TEHNO

Amenințarea GSM

PROFIL



John von Neumann

din sumarul numărului:

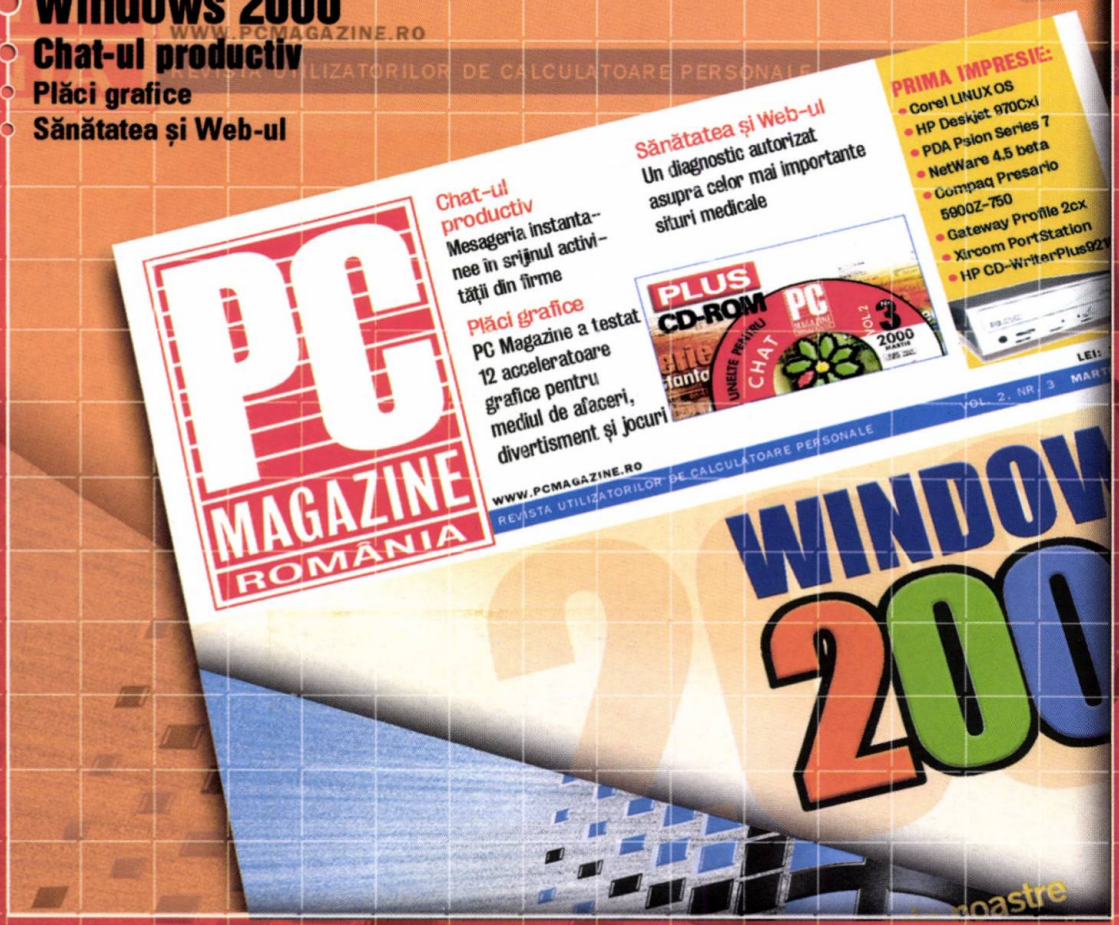
Programarea dinamică în problemele NP-complete

PC MAGAZINE ROMÂNIA

COMPUTER PRESS AGORA

Din acest număr:

- **Windows 2000**
- **Chat-ul productiv**
- **Plăci grafice**
- **Sănătatea și Web-ul**



www.pcmagazine.ro

Revista utilizatorilor de calculatoare personale

COMPUTER PRESS AGORA

Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

Vol. 10, Nr. 2/2000

gazeta de informatică

sumar

probleme.. PAG. 5

propuse pentru REZOLVARE

Olimpiada Internațională de Informatică

tehno.. PAG. 31

Amenințarea GSM
Mihai SCORȚARU

Efectele folosirii telefoanelor mobile

concurs.. PAG. 10

Rezultatele rundei a treia

Clasament

Problemele rundei a cincea

profii.. PAG. 34

John von NEUMANN
Crina GROȘAN

serial.. PAG. 18

Introducere în teoria GRAFURILOR
Cristian CADAR

Pagini WEB cu JavaScript
Mihai SCORȚARU

focus.. PAG. 37

PROGRAMAREA dinamică în problemele NP COMPLETE
Mihai OLTEAN

begin.. PAG. 26

ALGORITMI... la gura sobei
Sorin GROZA

bac.. PAG. 43

BACALAUREAT 2000

Al doilea bac în informatică

babel.. PAG. 29

Terminarea CURATĂ a unui program
Horea HAITONIC

dialog.. PAG. 48

POȘTA redacției

Răspunsuri la scrisori

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Cristian Cadar (București)
Sorin Groza (Aiud)
Horea Haitonic (Bistrița)
Mihai Oltean (Cugir)
Crina Groșan (Cluj)
Tehnoredactare:
Mihai Uricaru, Călin Borz
Corectura:
Carmen Bucur
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8, Cluj-Napoca, 3400
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Gabriela Bucșa (gbucsa@agora.ro)
Gabriel Piscupescu (gpiscup@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Anamaria Pascu (apascu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Mihaela Szabo (mszabo@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Se deschid noi FERESTRE

Pe data de 17 Februarie a avut loc lansarea oficială a noului sistem de operare *Microsoft Windows 2000*. Acest moment a marcat concretizarea celui mai important proiect software din istorie, cheltuielile de producție depășind suma de un miliard de dolari.

Potrivit oficialilor Microsoft, noul sistem de operare va produce o adevărată revoluție în domeniul IT, trecerea la noua versiune fiind comparabilă cu trecerea de la *Windows 3.1* la *Windows '95*. Conform așteptărilor, *Windows 2000* (*W2K*) oferă o serie de noi facilități, repară unele probleme apărute la versiunile anterioare și, mult mai important, introduce câteva noi tehnologii care își vor pune amprenta pe dezvoltarea viitoarelor proiecte software. Momentan *W2K* este disponibil în trei variante (*Professional*, *Server* și *Advanced Server*) fiecare adresându-se unui anumit sector de pe piață. Specialiștii afirmă că noul *Windows* este un sistem de operare mult mai bine realizat decât versiunile anterioare și că numărul problemelor care vor apărea va fi mult mai redus decât ne-am fi așteptat.

Imediat a apărut reacția adversarilor care au afirmat că, potrivit unui document intern al *Microsoft*, în codul sursă există peste 65.000 de probleme. Evident, majoritatea utilizatorilor s-au speriat la auzul acestei vești, dar lucrurile nu stau chiar așa. Adversarii s-au ferit să folosească termenul de *bug* deoarece acesta nu își avea rostul. În realitate problemele nu se referă la erori de funcționare ale sistemului ci la probleme apărute la scrierea codului, în sensul că anumite elemente nu au fost implementate atât de elegant cum s-ar fi dorit, dar acest lucru nu îi va afecta pe utilizatori. Neexistând nici un element care să poată fi criticat, a fost exploatat acest detaliu minor pentru ca să nu pară că *Microsoft* a realizat un sistem de operare perfect. Nu pot să afirm că *Windows 2000* se va comporta așa cum afirmă creatorii săi, inevitabilele erori vor apărea curând, dar trebuie spus faptul că noul sistem s-a comportat excelent începând chiar cu versiunile Beta, *Windows 2000 Release Candidate 2* fiind surprinzător de stabil.

Mă veți întreba ce legătură are această lansare cu învățământul românesc de informatică. Din păcate trebuie să recunosc că acesta va fi foarte puțin afectat de noul sistem de operare, în primul rând datorită lipsei dotării, dar, mult mai grav, din cauza celor care realizează programele școlare. În momentul în care în toată lumea se programează folosind medii vizuale, tinerii informaticieni români învață să programeze în *Pascal* (în *C* în cel mai bun caz) fără să știe să instaleze o imprimantă în *Windows '95*.

Nu mai poate mira pe nimeni faptul că majoritatea elevilor se plictisesc în timpul orelor de informatică în timp ce profesorul încearcă să le explice problema celor opt regine, drumul minim într-un graf sau alte subiecte la fel de "interesante". Cred că situația ar fi alta dacă elevul ar vedea că, efectuând câteva click-uri și scriind numai câteva linii de cod, realizează o întreagă aplicație.

La un moment dat, dl. ***Emil Onea***, președintele *Comisiei Naționale de Informatică* (*CNI*), spunea *Comunic, deci exist!* Dacă și restul comisiei ar gândi la fel și nu ar fi preocupați în primul rând de impunerea propriului punct de vedere, probabil că nu ar mai exista elevi de clasa a XII-a care, deși au studiat patru ani informatica, nu știu ce înseamnă *World Wide Web*, *e-mail* sau *browser*. Chiar dacă lipsa dotării nu permite în acest moment studiul ultimelor tehnologii IT, totuși nu trebuie ca elevii să învețe lucruri care sunt depășite de 10 ani, cum ar fi limbajul *Pascal*. Se spune că e prea complicat să se treacă de la *Pascal* la *C*, că profesorii nu sunt pregătiți etc.

Situația a fost similară acum câțiva ani când a avut loc trecerea de la *Basic* la *Pascal*. Și totuși profesorii s-au descurcat atunci chiar dacă acea trecere a fost mult mai dificilă deoarece s-a trecut de la programarea nestructurată la cea structurată, două principii de programare total diferite. Principiile care stau la baza programării în *C* sunt aceleași ca și cele ale programării în *Pascal*, iar limbajul este incomparabil mai puternic. În plus, după însușirea acestor principii, se poate trece foarte ușor la programarea obiectuală. De aici, pasul la mediile vizuale de dezvoltare nu este un "salt uriaș" și poate fi făcut, independent, de către cei interesați. Se pare însă că opinia *CNI* este diferită, membrii acesteia preferând să rămână cu 10 ani în urmă. Chiar dacă noua programă școlară nu specifică limbajul de programare care trebuie studiat, în principiu profesorii putând să nu aleagă *Pascal*-ul, totuși mai puțin de 10% dintre profesori optează pentru *C* și nu cred că cineva a avut curajul să aleagă *Java*.

Dacă aceste lucruri nu se vor schimba în curând, viitorul școlii românești de informatică poate fi pus sub semnul întrebării.

În concluzie, chiar dacă în lumea IT se deschid perspective noi, se pare că noi preferăm să ne întoarcem cu spatele. Poate că într-un viitor (sper apropiat) și tinerii români vor putea studia ultimele tehnologii dacă va exista voința necesară. Și aceasta nu depinde numai de *CNI*, pentru că aceasta nu poate rezolva problema dotării școlilor cu calculatoare performante.

Mihai Scorțaru

PROBLEME propuse pentru REZOLVARE

Olimpiada Internațională de Informatică a fost organizată anul acesta în orașul Antalya, în Turcia. Vă propunem să încercați să rezolvați problemele cu care s-au confruntat concurenții la această competiție.

P020001: La florărie

Dorești să-ți aranjezi florile în vitrina florăriei într-un mod cât mai plăcut. Ai F buchete de flori diferite, un buchet fiind compus din mai multe flori, având același tip și ai câteva vaze așezate într-o linie. Vazele sunt fixate pe un raft și sunt numerotate consecutiv de la 1 la V, unde V reprezintă numărul de vaze. Numărătoarea s-a făcut de la stânga la dreapta. Buchetele trebuie puse în vaze și sunt identificate în mod unic printr-un număr întreg de la 1 la F. Aceste numere de identificare au următoarea semnificație: ele determină ordinea în care trebuie să apară buchetele în vaze: buchetul i trebuie așezat într-o vază care se află la stânga față de vaza care conține buchetul j pentru oricare $i < j$. Să presupunem, de exemplu, că numărul de ordine al buchetului de azalee este 1, numărul de ordine al buchetului de begonii este 2 și numărul de ordine al buchetului de garoafe este 3. Toate buchetele trebuie așezate în vaze conform numerelor lor de ordine. Buchetul de azalee trebuie așezat într-o vază aflată în stânga vazei în care se pun begoniile, iar begoniile într-o vază din stânga garoafelor. Dacă există mai multe vaze decât buchete, anumite vaze vor rămâne goale. Într-o vază se va așeza un singur buchet.

Fiecare vază are o caracteristică distinctă, astfel, în urma așezării unui buchet într-o vază rezultă o valoare estetică, exprimată printr-un număr întreg. Aceste valori estetice sunt prezentate în tabelul de mai jos. Dacă o vază rămâne goală, valoarea estetică este 0.

		VAZE				
		1	2	3	4	5
Buchete	1 (azalee)	7	23	-5	-24	16
	2 (begonii)	5	21	-4	10	23
	3 (garoafe)	-21	5	-4	-20	20

Conform tabelului, azaleele vor arăta meseriaș în vaza 2, dar mizerabil în vaza 4.

În ideea de a obține cea mai plăcută imagine trebuie să maximizezi suma valorilor estetice ale aranjamentului în condițiile respectării cerinței cu privire la ordinea florilor. Dacă mai multe aranjamente au aceeași sumă maximă a valorilor estetice se acceptă oricare dintre acestea. În fișierul de ieșire se va scrie un singur aranjament.

Restricții

- $1 \le F \le 100$, unde F reprezintă numărul buchetelor de flori. Buchetele sunt numerotate de la 1 la F.
- $F \le V \le 100$, unde V reprezintă numărul vazelor.
- $-50 \le A_{ij} \le 50$, unde A_{ij} reprezintă valoarea estetică obținută dacă buchetul i se pune în vaza j.

Intrare

Numele fișierului de intrare este `FLOWER.INP`.

- pe prima linie sunt scrise două numere: F și V;
- fiecare din următoarele F linii conțin V numere întregi, astfel A_{ij} este al j-lea număr în cea de-a $(i+1)$-a linie a fișierului de intrare.

Ieșire

Fișierul de ieșire `FLOWER.OUT` trebuie să conțină două linii:

- Prima linie va conține suma valorilor estetice obținută pentru aranjamentul care se va descrie pe cea de-a doua linie.
- Cea de-a doua linie descrie aranjamentul sub forma unei secvențe de F numere, astfel încât al k-lea număr de pe această linie reprezintă numărul de ordine al vazei în care s-a așezat buchetul k.

Exemplu

`FLOWER.INP`

```
3 5
7 23 -5 -24 16
5 21 -4 10 23
-21 5 -4 -20 20
```

FLOWER.OUT
```
53
2 4 5
```

P020002: Coduri ascunse

Se consideră un anumit număr de cuvinte și un text. Textul conține un mesaj format din cuvintele care s-au ascuns în acesta într-o formă ciudată (și posibil ambiguă).

Cuvintele și textul sunt secvențe de litere mici și majuscule aparținând doar alfabetului englez. Literele mici diferă de majuscule. Lungimea cuvântului este definită în modul cunoscut. De exemplu, cuvântul ALL are lungimea 3.

Literele cuvântului nu trebuie să apară consecutiv în textul dat. De exemplu, cuvântul ALL va apărea în text întotdeauna în interiorul unei subsecvențe de forma A*u*L*v*L unde *u* și *v* sunt secvențe arbitrare de litere (posibil vide). Spunem că A*u*L*v*L este *o secvență acoperitoare* pentru ALL. În general, *o secvență acoperitoare* corespunzătoare unui cuvânt se definește ca fiind o subsecvență a textului, astfel încât prima și ultima literă a subsecvenței sunt identice cu prima și ultima literă ale cuvântului, iar prin ștergerea anumitor litere (sau a nici uneia) este posibil să se obțină cuvântul. Trebuie să rețineți că un cuvânt poate să apară în interiorul uneia sau a mai multor secvențe acoperitoare sau să nu apară deloc în text. De asemenea, rețineți că o secvență acoperitoare poate fi secvență acoperitoare pentru mai multe cuvinte.

O secvență acoperitoare se identifică prin numărul de ordine (în cadrul textului) al primei litere și prin numărul de ordine al ultimei litere. (Prima literă din text are numărul de ordine 1.) Spunem că două secvențe acoperitoare c_1 și c_2, *nu se suprapun* dacă prima literă din c_1 are numărul de ordine mai mare (>) decât ultima literă din c_2 sau invers. În caz contrar, spunem că cele două secvențe acoperitoare se *suprapun*.

În scopul extragerii mesajului ascuns în text, trebuie să construiești o *soluție*. O *soluție* este formată din mai multe seturi, care la rândul lor sunt formate fiecare dintr-un cuvânt și dintr-o secvență acoperitoare a cuvântului respectiv, respectând următoarele condiții:

a) Secvențele acoperitoare nu se suprapun.
b) Lungimea maximă a unei secvențe acoperitoare este 1000.
c) Suma lungimilor cuvintelor este maximă. (Numărul termenilor din suma cerută este egal cu numărul seturilor, în suma respectivă aflându-se lungimile cuvintelor din fiecare set.)

În cazul în care există mai multe soluții, se cere una singură.

Restricții

- $1 \le N \le 100$ unde N reprezintă numărul cuvintelor.
- Lungimea maximă a unui cuvânt este 100 de litere.
- $1 \le$ lungimea textului dat ≤ 1000000 litere.

Spunem despre o secvență acoperitoare *c*, corespunzătoare unui cuvânt *w*, că este *dreapta-minimală* dacă nici un prefix propriu al lui *c* nu este secvență acoperitoare pentru *w*. (Un *prefix propriu* al lui *c* este o subsecvență de început a lui *c* care nu este egală cu *c*.) De exemplu, pentru cuvântul ALL, secvența acoperitoare AAALAL este *dreapta-minimală*, pe când secvența AAALALAL este acoperitoare, dar nu este *dreapta-minimală*.

Se garantează că

a) pentru fiecare poziție din textul dat, numărul secvențelor acoperitoare *dreapta-minimale*, care evident se vor suprapune, este cel mult 2500;
b) numărul secvențelor acoperitoare *dreapta-minimale* este cel mult 10000.

Intrare

Vor exista două fișiere text: **WORDS.INP** și **TEXT.INP**. Fișierul **WORDS.INP** conține lista cuvintelor, iar **TEXT.INP** conține textul.

- Prima linie a fișierului **WORDS.INP** conține numărul *N*.
- Fiecare din următoarele *N* linii conține câte un cuvânt care este scris sub forma unei secvențe de litere, fără spații. Cuvintele se identifica prin numerele lor de ordine din fișierul de intrare **WORDS.INP**; acestea sunt numere întregi având valori între 1 și *N*.
- Fișierul **TEXT.INP** conține o secvență de litere (la sfârșitul căreia există simbolurile *sfârșit de linie* și *sfârșit de fișier*). Fișierul nu conține spații.

Ieșire

Fișierul de ieșire de tip text va avea numele **CODES.OUT**.

- Pe prima linie se va scrie suma obținută pe baza soluției obținute.
- Fiecare din următoarele linii descrie un set al soluției: o linie va conține trei numere întregi *i*, *s*, *e*, despărțite prin câte un spațiu. (*i* este numarul de ordine al cuvântului care apare în secvența acoperitoare descrisă prin *s* și *e*, unde *s* este numărul de ordine în text al primei litere a secvenței acoperitoare, iar *e* reprezintă numărul de ordine în text al ultimei litere a secvenței acoperitoare.) Ordinea acestor linii în fișier este oarecare.

Exemplu

WORDS.INP
```
4
RuN
RaBbit
HoBbit
StoP
```

TEXT.INP
```
StXRuYNvRuHoaBbvizXztNwRRuuNNP
```

CODES.OUT
```
12
2 9 21
1 4 7
1 24 28
```

Explicație

Mesajul ascuns care poate fi extras din soluție este "`RuN RaBbit RuN`". (O altă soluție ar putea fi "`RuN HoBbit RuN`").

P020003: Oraș subteran

Să presupunem că ești prizionerul unui oraș subteran din Cappadocia. Presupunem în continuare că ai avut un noroc incredibil și ai găsit harta orașului. Din nefericire pe ea nu este marcat punctul unde te afli. Trebuie să determini locul, unde ai găsit harta explorând orașul.

Harta orașului este de formă dreptunghiulară, împărțită în pătrate având latura unitate. Fiecare pătrat poate fi parte dintr-un coridor sau parte dintr-un zid. Coridorul este marcat cu litera 'O', zidul este marcat cu litera 'W'. Direcția nord este marcată pe hartă. Din fericire ai o busolă astfel încât poți orienta corect harta. Inițial, te afli în coridor.

Totul începe prin apelarea procedurii (sau funcției) `start` fără argumente. Poți să explorezi orașul utilizând procedurile (sau funcțiile) `look` și `move`.

Poți pune întrebări apelând funcția `look(`*dir*`)` unde *dir* arată direcția spre care privești, care poate fi exprimată prin unul din caracterele 'N', 'S', 'E' și 'W' reprezentând nord, sud, est și respectiv vest. Presupunem că valoarea argumentului *dir* este 'N'. Dacă pătratul vecin, spre nord, cu pătratul pe care te afli este parte din coridor, răspunsul va fi litera 'O' iar în cazul în care acesta este parte din zid, răspunsul va fi 'W'. Similar, este posibil să privești și să primești informație despre alte pătrate învecinate celui în care te afli.

Poți să treci într-unul din cele patru pătrate vecine, folosind `move(`*dir*`)` unde *dir* reprezintă direcția pasului și a fost prezentată anterior. Poți să te muți doar într-un pătrat ce reprezintă o parte din coridor. Încercarea de a te muta într-un pătrat care reprezintă o parte dintr-un zid, va fi o greșală gravă. Este posibil să ajungi în orice pătrat reprezentând coridor, pornind dintr-un loc care de asemenea este parte din coridor.

Trebuie să găsești poziția pătratului unde ai găsit harta, apelând `look(`*dir*`)` de un număr minim de ori. Odată ce poziția hărții a fost determinată, aceasta trebuie înregistrată apelând `finish(`*x*`,`*y*`)` unde *x* reprezintă coordonata orizontală (vest-est) iar *y* reprezintă coordonata verticală (sud-nord) a poziției.

Restricții

- $3 \le U \le 100$, unde U este lățimea hărții, adică lungimea hărții în număr de pătrate pe orizontală (vest-est);
- $3 \le V \le 100$, unde V este înălțimea hărții, adică lungimea hărții în număr de pătrate pe verticală (sud-nord);
- orașul este încercuit de ziduri incluse pe hartă;
- colțul de sud-vest al orașului are coordonatele (1,1) și colțul de nord-est are coordonatele (U,V).

Intrare

Numele fișierului de intrare de tip text este **`UNDER.INP`**.

- Prima linie conține două numere: U, V.
- Fiecare din următoarele V linii conțin o linie a hărții pe direcția orizontală. Fiecare linie conține U caractere, astfel încât al x-lea caracter de pe a (V-y+2)-a linie din fișierul de intrare conține informație despre poziția (x,y) a hărții: litera 'W' reprezintă zid, litera 'O' reprezintă coridor. Informațiile de pe linii nu sunt despărțite prin spații.

Ieșire

Nu va fi generat nici un fișier de ieșire. Rezultatul găsit de programul tău trebuie să fie raportat apelând subprogramul `finish(`*x*`,`*y*`)`.

Exemplu

`UNDER.INP`:

```
5 8
WWWWW
WWWOW
WWWOW
WOOOW
WOWOW
WOOWW
WWOOW
WWWWW
```

O posibilă secvență care se încheie cu apelul corect al subprogramului `finish`:

Interacțiunea

```
start ()
look('N')    'W'
look('E')    'O'
move('E')
look('E')    'W'
finish(3,5)
```

Instrucțiuni pentru programatorii în Pascal

Programul vostru trebuie să conțină linia:

```
uses undertpu;
```

Acest unit va furniza următoarele:

```
procedure start;              {apelată prima}
function look (dir:Char):Char;
procedure move (dir:Char);
procedure finish (x,y:Integer);
                              {apelată ultima}
```

Instrucțiuni pentru programatorii în C/C++

Programul vostru trebuie să conțină linia:

```
#include "under.h"
```

Acesta va furniza următoarele:

```
void start (void);          // apelată prima
char look (char);
void move (char);
void finish (int,int); // apelată ultima
```

P020004: Semafoare

În orașul Dingilville traficul este organizat într-un mod neobișnuit. Există intersecții și drumuri care leagă aceste intersecții. Între două intersecții există cel mult un drum. Nu există drum care să lege o intersecție cu ea însăși. Timpul de deplasare pe un drum este același pentru ambele

direcții. La fiecare intersecție este un singur semafor a cărui culoare într-un anumit moment este fie albastră fie roșie. Culoarea fiecărui semafor alternează periodic: o anumită perioadă de timp culoarea este albastră, o altă perioadă este roșie. Traficul permite deplasarea între oricare două intersecții, dacă și numai dacă, semafoarele din ambele intersecții sunt de aceeași culoare în momentul plecării din una din ele spre cealaltă. Dacă un vehicul sosește într-o intersecție exact în momentul în care cel puțin una din culorile semafoarelor se schimbă, trebuie luate în considerare culorile noi ale semafoarelor. Vehiculele pot aștepta în intersecții. Se dă o hartă a drașului pe care sunt afișate:

- timpii de deplasare pentru toate drumurile (numere întregi);
- duratele celor două culori corespunzătoare fiecărei intersecții (numere întregi);
- culoarea inițială a semaforului și timpul rămas (număr întreg) pentru această culoare până la schimbarea acestei culori pentru fiecare intersecție.

Găsiți modul de parcurgere a drumului (după ce a pornit traficul) dintre punctul de pornire și punctul de sosire, astfel încât timpul de parcurgere să fie minim. În cazul în care există mai multe drumuri care satisfac cerințele problemei se va afișa unul singur.

Restricții

- $2 \le N \le 300$, unde N reprezintă numărul de intersecții. Intersecțiile sunt identificate prin numere întregi de la 1 la N. Aceste numere se numesc numere de identificare.
- $1 \le M \le 14000$, unde M reprezintă numărul drumurilor.
- $1 \le l_{ij} \le 100$, unde l_{ij} reprezintă timpul necesar de deplasare de la intersecția i la intersecția j utilizând drumul de legătură dintre i și j.
- $1 \le t_{ic} \le 100$, unde t_{ic} este durata unei culori c pentru semaforul din intersecția i. Indexul c este B pentru culoarea albastră, respectiv P pentru culoarea roșie.
- $1 \le r_{ic} \le t_{ic}$, unde r_{ic} este timpul rămas pentru culoarea inițială c în intersecția i.

Intrare

Fișierul text de intrare este `LIGHTS.INP`:

- prima linie conține două numere: numărul de identificare al intersecției de plecare și numărul de identificare al intersecției de sosire;
- a doua linie conține două numere: N și M;
- următoarele N linii conțin informații despre cele N intersecții; pe linia $(i+2)$ a fișierului de intrare se află informații despre intersecția i : C_i, r_{ic}, t_{iB}, t_{iP} unde C_i este fie 'B' fie 'P', indicând culoarea inițială a semaforului din intersecția i;
- următoarele M linii conțin informații despre cele M drumuri. Fiecare linie are următorul format: i, j, l_{ij} unde i și j sunt numerele de identificare ale intersecțiilor care sunt legate prin acest drum.

Ieșire

Fișierul text de ieșire se numește `LIGHTS.OUT`

Dacă există drum, atunci:

- prima linie va conține timpul minim de parcurgere a drumului dintre intersecția de pornire și intersecția de sosire.
- A doua linie va conține lista intersecțiilor de pe drumul minim găsit. În fișierul de ieșire trebuie scrise intersecțiile în ordinea parcurgerii lor. Astfel, primul număr întreg din această linie va fi numărul de identificare pentru intersecția sursă și ultimul număr întreg va fi numărul de identificare a intersecției destinație.

Dacă nu există drum, atunci fișierul va conține o singură linie pe care se va scrie numărul întreg 0.

Exemplu

LIGHTS.INP	LIGHTS.OUT
1 4	127
4 5	1 2 4
B 2 16 99	
P 6 32 13	
P 2 87 4	
P 38 96 49	
1 2 4	
1 3 40	
2 3 75	
2 4 76	
3 4 77	

P02005: Nivelare

Fie un joc care folosește un șir de N stive unde fiecare stivă conține zero sau mai multe discuri. Vezi figura 1. Stivele sunt identificate prin numere întregi de la 1 la N. O mutare în acest joc constă din alegerea unei stive (p) și specificarea unui număr (k) urmând să se mute câte k discuri de pe stiva p pe stivele vecine ale acesteia. Vezi figura 2. Stiva p are doi vecini, p-1 și p+1 dacă $1<p<N$; stiva 1 (p=1) are un singur vecin: stiva 2; stiva N (p=N) are de asemenea un singur vecin: stiva N-1. Rețineți că pentru a putea realiza o astfel de mutare, pe stiva p trebuie să existe cel puțin $2k$ discuri dacă stiva p are doi vecini și cel puțin k discuri, dacă stiva p are un singur vecin.

Scopul jocului este de a "nivela", printr-un număr minim de mutări, toate stivele, ceea ce înseamnă că cele N stive trebuie să conțină același număr de discuri. În cazul în care există mai multe soluții, se va afișa una singură.

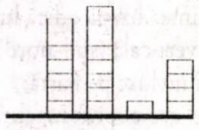


Fig. 1: Cinci stive cu 0, 7, 8, 1 și 4 discuri

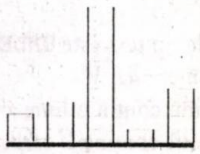


Fig. 2: Aceleași stive după o mutare: p=2, k=2

Restricții

- se garantează că stivele se pot nivela în cel mult 10000 de mutări;
- $2 \le N \le 200$;
- $0 \le C_i \le 2000$, unde C_i este numărul discurilor de pe stiva i la începutul jocului ($1 \le i \le N$).

Intrare

Fișierul de intrare **FLAT.INP** are două linii:

- pe prima linie se află N;
- pe a doua linie se află N numere întregi; al i-lea număr va fi valoarea lui C_i.

Ieșire

Fișierul text de ieșire va fi **FLAT.OUT**:

- prima linie va conține numărul de mutări (fie acesta M);
- următoarele M linii vor conține două numere întregi reprezentând mutarea: p, k.

Ordinea mutărilor în fișierul de ieșire trebuie să coincidă cu ordinea în care s-au făcut mutările în timpul jocului. Astfel, prima mutare trebuie să fie scrisă în a doua linie a fișierului de ieșire.

Exemplu

FLAT.INP

```
5
0 7 8 1 4
```

FLAT.OUT

```
5
5 2
3 4
2 4
3 1
4 2
```

P020006: Teren

Locuitorii din Dingilville caută un teren pentru construirea unui aeroport, având acces la harta regiunii. Această hartă are formă dreptunghiulară și este împărțită în pătrate având latura unitate. Fiecare pătrat se identifică printr-o pereche de coordonate (x,y), unde x este coordonata orizontală (vest-est), iar y este coordonata verticală (sud-nord). Pe hartă este indicată altitudinea fiecărui pătrat.

Găsiți o regiune dreptunghiulară de pătrate având aria cea mai mare (adică un teren dreptunghiular format din cel mai mare număr de pătrate) astfel încât:

a) cea mai mare diferență dintre pătratul cu cea mai mare altitudine și pătratul cu cea mai mică altitudine de pe teren este mai mic sau egal cu o valoare limită dată C;
b) lățimea (măsurată de-a lungul direcției vest-est) terenului este cel mult 100.

În cazul în care există mai multe terenuri ce satisfac cerințele problemei, se va afișa unul singur.

Restricții

- $1 \le U \le 700$, $1 \le V \le 700$, unde U și V reprezintă dimensiunile hărții. U reprezintă numărul pătratelor pe direcția vest-est, iar V reprezintă numărul pătratelor pe direcția sud-nord;
- $0 \le C \le 10$;
- $-30{,}000 \le H_{xy} \le 30{,}000$, unde numărul întreg H_{xy} este altitudinea pătratului având coordonatele (x, y), $1 \le x \le U$, $1 \le y \le V$;
- colțul sud-vest al hărții are coordonatele (1,1) iar colțul nord-est are coordonatele (U,V).

Intrare

Numele fișierului de intrare este **LAND.INP**:

- pe prima linie se află trei numere întregi: U, V și C;
- fiecare din următoarele V linii conține numerele întregi H_{xy} pentru $x = 1,\ldots,U$. Pe cea de-a $(V-y+2)$-a linie H_{xy} aparține celui de-al x-lea număr.

Ieșire

Fișierul de ieșire **LAND.OUT** va conține o linie pe care se vor scrie patru numere întregi, localizând terenul determinat: X_{min}, Y_{min}, X_{max}, Y_{max}, unde (X_{min}, Y_{min}) sunt coordonatele pătratului din colțul sud-vest al terenului, iar (X_{max}, Y_{max}) sunt coordonatele pătratului din colțul nord-est al terenului.

Exemplu

LAND.INP

```
10 15 4
41 40 41 38 39 39 40 42 40 40
39 40 43 40 36 37 35 39 42 42
44 41 39 40 38 40 41 38 35 37
38 38 33 39 36 37 32 36 38 40
39 40 39 39 39 40 40 41 43 41
39 40 41 38 39 38 39 39 39 42
36 39 39 39 39 40 39 41 40 41
31 37 36 41 41 40 39 41 40 40
40 40 40 42 41 40 39 39 39 39
42 40 44 40 38 40 39 39 37 41
41 41 40 39 39 40 41 40 39 40
47 45 49 43 43 41 41 40 39 42
42 41 41 39 40 39 42 40 42 42
41 44 49 43 46 41 42 41 42 42
45 40 42 42 46 42 44 40 42 41
```

LAND.OUT

```
4 5 8 11
```

XI. INTERNATIONAL

OLYMPIAD in INFORMATICS

CONCURS

A luat sfârșit și cea de-a treia etapă a concursului Bursele Agora, organizat de Gazeta de Informatică. Vă prezentăm acum clasamentul acestei runde, clasamentul intermediar după trei etape, precum și soluțiile problemelor propuse spre rezolvare în cadrul rundei a treia. De asemenea, tot aici veți găsi și enunțurile problemelor de la runda a cincea.

Precizare...

La prima problemă de la runda a treia apărea o contradicție pentru ordinea citirii numerelor m și n.

În cadrul enunțului ordinea era m, n, iar în cadrul exemplului ordinea era n, m.

Evident, am considerat corecte ambele variante.

Specificații...

Suntem nevoiți din nou să vă rugăm să respectați cu strictețe specificațiile problemelor. Dacă nu se precizează altfel, fișierele de intrare se află în directorul curent, iar fișierele de ieșire trebuie să fie în același director. De asemenea, denumirile fișierelor de intrare și de ieșire trebuie neapărat să fie cele indicate în enunțuri, în caz contrar concurentul nu primește nici un punct pentru problema respectivă, deoarece programul de evaluare verifică numai fișierul al cărui nume este identic cu cel al fișierului de ieșire, precizat în enunț.

Unit-ul CRT...

Rugăm din nou programatorii în Pascal să nu mai folosească unit-ul `crt`. Reamintim faptul că evaluarea are loc pe un calculator dotat cu procesor *Intel Pentium III*, deci compilând programul folosind uneltele puse la dispoziție de firma *Borland/Inprise*, va apărea inevitabil eroarea de execuție `Error 200: Division by 0`. Nici una dintre problemele propuse în cadrul concursului Bursele Agora nu necesită folosirea acestui unit.

Termen...

Rezolvările problemelor publicate în acest număr pot fi trimise până la data de

31 Martie 2000

la **Computer Libris Agora**,
Universității 8/8, Cluj-Napoca, 3400
sau e-mail: office@ginfo.ro

Runda 3 Top 20

1.	Daniel Dumitran,București	244	
2.	Adrian Cârcu, Bistrița	240	
3.	Gabriel Erzse, Oradea	220	
4.	Cristian Cadar, București	218	
5.	Mihai Pătrașcu, Craiova	200	
6.	Mihai Stroe, București	199	
7.	Robert Baban, Baia Mare	192	
8.	Cosmin Silvestru Negruseri, Bistrița	182	
8.	Ciprian Baicu, Dr. Turnu-Severin	182	
10.	Silviu Bota, Baia Mare	177	
11.	Ioana Ileană, Alba Iulia	175	
12.	Vlad Vâlceanu, Arad	170	
13.	Cristian Alexandrescu, Botoșani	145	
14.	David Andrei, Râmnicu Vâlcea	130	
15.	Virgil Șerbănuță, București	110	
16.	Mihai Dondera, București	106	
17.	Sorin Otescu, Brașov	105	
18.	Costin Speciac, Buzău	100	
19.	Marian Anghel, Buftea	98	
20.	Csaba Patcas, Oradea	97	

Clasament GInfo Top 50

1	Adrian Cârcu, Bistrița	828
2	Mihai Pătrașcu ,Craiova	771
3	Cristian Cadar, București	763
4	Ioana Ileană, Alba Iulia	746
5	Gabriel Erzse, Oradea	743
6	Daniel Dumitran, București	724
7	Vlad Vâlceanu, Arad	684
8	Silviu Bota, Baia Mare	665
9	Cosmin Șilvestru Negruseri, Bistrița	602
10	Csaba Patcas, Oradea	593

11.	Ciprian Baicu, Dr. Turnu-Severin	570	
12.	Mihai Stroe, București	569	
13.	Cristian Alexandrescu, Botoșani	529	
14.	Virgil Șerbănuță, București	510	
15.	David Andrei, Râmnicu Vâlcea	481	
16.	Marian Anghel, Buftea	454	
17.	Ileana Constantinescu, Sibiu	448	
18.	Lucian Burja, Blaj	430	
19.	Horea-Adrian Haitonic, Năsăud	418	
20.	Mihai Dondera, București	411	
21.	Adrian Sándor, București	405	
22.	Mihai Maerean, Slobozia	396	
23.	Robert Baban, Baia Mare	392	
24.	Marius Muja, Orăștie	386	
25.	Pal Szasz, Marghita	380	
25.	Marius George Andrei, București	380	
27.	Adrian Bălan, Târgu Jiu	370	
28.	Valentin Muste, Baia Mare	368	
29.	Sorin Otescu, Brașov	346	
30.	Costin Speciac, Buzău	343	
31.	Ciprian Cană, Vaslui	337	
32.	Ionuț Trifan, Buzău	335	
33.	Codruț-Lucian Lazăr, Câmpeni	326	
34.	Gabriel Pârvan, Râmnicu Sărat	325	
35.	Daniel Popa, Orăștie	323	
36.	Marius Neagu, București	318	
37.	Victor Asavei, Dr. Turnu-Severin	311	
38.	Arcadie Crăcan, Bălți	310	
39.	Robert Bădiță, Craiova	309	
40	Daniel Bolohan, Iași	306	
41.	Liviu Păunescu, Constanța	301	
42.	George Pârcăbălescu, Slatina	297	
43.	Horea Coroiu, Cluj-Napoca	293	
44.	Marius Botiș, Baia Mare	289	
45.	Ion Nechita, Slatina	287	
46.	George Alexandru Cocu, Galați	285	
47.	Andrei Mihnea Răileanu, Slatina	273	
48.	Vlad Dascălu, Bacău	273	
49.	Paul Rusu, Cluj-Napoca	271	
50.	Liviu Andreicuț, ***	268	

Runda 3 Problema 1

Blocaje (ruperea conexității)

Problema se reduce la determinarea numărului minim de muchii care trebuie eliminate pentru ca un graf neorientat să nu fie conex. Punctajul mediu obținut de concurenți la această problemă a fost de 43,97 puncte, o medie acceptabilă având în vedere dificultatea destul de ridicată a problemei.

Mulți concurenți au fost dezorientați de faptul că în enunț nu erau precizate limite pentru datele de intrare. De data aceasta nu a fost o greșeală a redacției, omisiunea fiind intenționată, deoarece la multe concursuri (printre care și ACM) asemenea limite nu sunt precizate întotdeauna.

Datele de test au fost "la limita bunului simț", neexistând cazuri în care numărul de orașe să depășească 100, deci nu ar fi trebuit să apară probleme la memorarea grafului.

Vă prezentăm în continuare soluția propusă de *Cristian Alexandrescu*, din Botoșani.

Având două noduri diferite *s* și *t* ale grafului, pentru a le deconecta trebuie să înlăturăm un număr minim de muchii care este egal cu numărul maxim de drumuri disjuncte (ca muchii) dintre *s* și *t* (teorema este asemănătoare cu teorema lui *Menger,* demonstrația fiind similară dacă se folosește teorema flux maxim/tăietură de capacitate minimă: *Într-o rețea de transport tăietura minimă este egală cu fluxul maxim.*).

Pentru determinarea numărului de drumuri disjuncte (ca muchii) dintre *s* și *t,* se folosește algoritmul *Edmonds-Karp,* aplicat pe o rețea de transport, obținută prin înlocuirea fiecărei muchii a grafului inițial printr-o pereche de arce simetrice, fiecare dintre aceste arce având capacitatea 1. Un flux de valoare maximă, definit pe această rețea, reprezintă numărul maxim de drumuri disjuncte (ca muchii) dintre *s* și *t*. Aparent, ar trebui să aplicăm algoritmul pentru flux maxim pentru oricare două noduri *i* și *j* ale grafului și să alegem valoarea minimă dintre toate valorile obținute pentru fluxul maxim. De fapt, este suficientă aplicarea algoritmului *Edmonds-Karp* de *n*-1 ori pentru rețeaua de transport având intrarea *s* un nod oarecare (fixat) iar ieșirea (*t*) fiind orice nod din rețeaua de transport diferit de *s*. Acest lucru este dovedit de faptul că, înlăturând oricare muchii care îndeplinesc condiția din enunț (că deconectează cel puțin două vârfuri) va fi deconectată intrarea *s* de cel puțin un vârf care, la un moment dat va fi ieșire pentru algoritmul de determinare a fluxului maxim.

Muchiile care trebuie înlăturate reprezintă o tăietură de valoare minimă în rețeaua pentru care s-a obținut valoarea minimă a fluxului maxim.

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
David Andrei, Râmnicu Vâlcea
Adrian Cârcu, Bistrița
Mihai Dondera, București
Cosmin Silvestru Negruseri, Bistrița
Mihai Pătrașcu, Craiova
Costin Speciac, Buzău
Vlad Vâlceanu, Arad

Deoarece algoritmul *Edmonds-Karp este* aplicat de *n*-1 ori, ordinul de complexitate al algoritmului prezentat anterior poate fi obținut adăugând factorul *n* la ordinul de complexitate al algoritmului *Edmonds-Karp* ($O(m^2 \cdot n)$). Obținem astfel un ordin de complexitate $O(m^2 \cdot n^2)$.

Algoritmul descris a fost implementat astfel:

Listing BLOCAJE.PAS

```
Program blocaje;
const max=250;
      cevamare=255;
type ref=^structura;
     structura=record
                 nod:Byte;
                 next:ref
               end;
     mref=^mstructura;
     mstructura=record
                  nod1,nod2:Byte;
                  next:mref
                end;
     calemarita=array[1..max]of Byte;
     flux=array[1..max,1..max]of Byte;
var l:array[1..max]of ref;
     n,m,s,t:Integer; solflux:Byte;
     sol:mref;

procedure citire;
var fin:Text; aux:ref;
     i:Integer; v1,v2:Byte;
     d:array[1..max]of Byte;
begin
  Fillchar(d,Sizeof(d),0);
  Assign(fin,'BLOCAJ.IN'); Reset(fin);
  Readln(fin,n,m);
  for i:=1 to n do l[i]:=nil;
```

```
  for i:=1 to m do
    begin
      Readln(fin,v1,v2);
      Inc(d[v1]); Inc(d[v2]);
      New(aux); aux^.nod:=v1;
      aux^.next:=l[v2]; l[v2]:=aux;
      New(aux); aux^.nod:=v2;
      aux^.next:=l[v1]; l[v1]:=aux
    end;
  s:=1;
  for i:=2 to n do
    if d[i]<=d[s] then s:=i;
  solflux:=d[s]+1;
  Close(fin)
end;

function gasestecalemarita(var f:flux;
  var dr:calemarita; var lung:Byte):Boolean;
var tata,coada:array[1..max]of Byte;
    q1,q2,v1,v2:Byte; p1:ref;
begin
  Fillchar(tata,Sizeof(tata),0);
  tata[s]:=cevamare;
  coada[1]:=s; q1:=1; q2:=1;
  while (tata[t]=0) and (q1<=q2) do
    begin
      v1:=coada[q1]; p1:=l[v1];
      while p1<>nil do
        begin
          v2:=p1^.nod;
          if (tata[v2]=0) and ((f[v1,v2]=0)
                      or (f[v2,v1]=1)) then
          begin
            Inc(q2);
            coada[q2]:=v2; tata[v2]:=v1
          end;
          p1:=p1^.next
        end;
      Inc(q1)
    end;
  if tata[t]<>0 then
    begin
      gasestecalemarita:=true;
      lung:=0; v1:=t;
      while v1<>cevamare do
        begin Inc(lung);
          dr[lung]:=v1; v1:=tata[v1]
        end
    end
  else gasestecalemarita:=false
end;

procedure aflataietura(var f:flux);
var tata,coada:array[1..max]of Byte;
    q1,q2,v1,v2:Byte; p1:ref;
    maux:mref;
begin
  fillchar(tata,sizeof(tata),0);
  tata[s]:=1; coada[1]:=s;
  q1:=1; q2:=1;
  while q1<=q2 do
    begin
      v1:=coada[q1];
      p1:=l[v1];
      while p1<>nil do
        begin
          v2:=p1^.nod;
          if (tata[v2]=0) and ((f[v1,v2]=0)
                      or (f[v2,v1]=1)) then
          begin
            Inc(q2);
            coada[q2]:=v2; tata[v2]:=1
          end;
          p1:=p1^.next
        end;
      Inc(q1)
    end;
  sol:=nil;
  for v1:=1 to n do
    if tata[v1]=1 then
    begin
      p1:=l[v1];
      while p1<>nil do
        begin
          if (tata[p1^.nod]=0) and
             (f[v1,p1^.nod]=1) and
             (f[p1^.nod,v1]=0) then
          begin
            New(maux); maux^.nod1:=v1;
            maux^.nod2:=p1^.nod;
            maux^.next:=sol; sol:=maux
          end;
          p1:=p1^.next
        end
    end
end;

procedure edmonds_karp;
var f:flux; dr:calemarita;
    lung,i,v1,v2,maxflux:Byte;
begin
  Fillchar(f,Sizeof(f),0); maxflux:=0;
  while gasestecalemarita(f,dr,lung) do
    begin
      for i:=lung downto 2 do
        begin
          v1:=dr[i]; v2:=dr[i-1];
          if f[v1,v2]=0 then f[v1,v2]:=1
          else f[v2,v1]:=0
        end;
      Inc(maxflux)
    end;
  if maxflux<solflux then
  begin solflux:=maxflux; aflataietura(f) end
end;

procedure rez;
var loop:Byte; fout:Text;
begin
  for loop:=1 to n do
    if(loop<>s) then
    else begin t:=loop; edmonds_karp end;
  Assign(fout,'BLOCAJ.OUT'); Rewrite(fout);
  Writeln(fout,solflux);
  while sol<>nil do
    begin
      writeln(fout,sol^.nod1,' ',sol^.nod2);
      sol:=sol^.next
    end;
  Close(fout)
end;

Begin
  citire; rez
End.
```

Runda 3 Problema 2

Numere prime

A fost aceasta cea mai dificilă problemă? Nici unul dintre concurenți nu a reușit să obțină mai mult de 80 de puncte; punctajul mediu a fost și el destul de mic și anume 35.16 puncte.

Au existat cel puțin două motive pentru care a fost propusă această problemă. În primul rând, în numărul 7/99 al GInfo a apărut un articol care trata verificarea primalității numerelor mari, deci am vrut să vedem dacă ați citit articolul. De asemenea, am dorit să încurajăm implementarea algoritmilor folosind limbajul *Java*, limbaj care oferă (prin intermediul metodei `isProbablePrime()` a clasei `BigInteger`) posibilitatea testării directe a primalității numerelor mari.

Din nefericire, puținii concurenți care au ales să folosească limbajul *Java* au comis unele erori și punctajele obținute au fost surprinzător de mici.

Majoritatea programatorilor în *C* și *Pascal* au rezolvat corect problema din punct de vedere logic, dar algoritmii nu au fost suficient optimizați și pentru numere foarte mari programele au depășit limita de timp admisă.

O altă eroare frecvent întâlnită a fost verificarea primalității numerelor `0` și `1` pe care mulți concurenți le-au considerat prime, chiar dacă din exemplul publicat reieșea clar că numărul `1` nu este prim.

Nu vă putem prezenta soluția propusă de un anumit concurent, deoarece nu a existat nici o rezolvare corectă. Din aceste motive am decis să publicăm soluția oficială care a fost implementată folosind limbajul *Java*.

Metoda `isProbablePrime()` a clasei `BigInteger` are un argument care indică numărul de teste efectuate pentru verificarea primalității numărului. Fiecare test are o probabilitate de eroare de `1/2`. Efectuând mai multe teste, probabilitatea de eroare scade exponențial, pentru `n` teste, ea devenind $1/2^n$.

Datorită faptului că programele sunt rulate pe un calculator *Intel Pentium III* cu frecvența de 450 MHz, valoarea *n* aleasă poate fi destul de mare, chiar și efectuarea a 100 de verificări încadrându-se în limita de timp (10 secunde/test - 1 secundă/număr). Totuși, chiar și 10 verificări ar fi fost suficiente, probabilitatea de eroare în acest caz fiind mai mică de 1‰.

Listing PRIME.JAVA

```
import java.math.*;
import java.io.*;
import java.lang.*;
```

CR3P2 Top 10

Robert Baban, Baia Mare	80
Ciprian Baicu, Dr. Tn. Severin	80
Arcadie Crăcan, Bălți	80
Gabriel Erzse, Oradea	80
Ioana Ileană, Alba Iulia	80
Silviu Bota, Baia Mare	70
Cristian Cadar, București	70
Daniel Dumitran, București	70
Sorin Otescu, Brașov	70
Vlad Vâlceanu, Arad	70

```
class prime{
  public static void main
                        (String args[]){
    try{
      RandomAccessFile input=
        new RandomAccessFile
                        ("PRIME.IN","r");
      PrintStream output=
        new PrintStream(
        new FileOutputStream
                         ("PRIME.OUT"));
      for (int i=0;i<10;i++) {
        String ss=input.readLine();
        if (ss==null)
          throw new IOException();
        // eliminarea caracterului
        // de sfarsit de linie
        ss=ss.substring(0,
                        ss.length()-1);
        BigInteger nr=
          new BigInteger(ss);
        // testarea primalitatii cu
        // o probabilitate de eroare
        // 1/2^100
        if (nr.isProbablePrime(100))
          output.println("DA");
        else output.println("NU");
      }
    } catch (IOException e) {};
  }
}
```

Runda 3 Problema 3

JOC

Din reacțiile concurenților rezultă că, probabil aceasta a fost cea mai interesantă problemă propusă până în prezent la concursul Bursele Agora; au existat și participanți care au afirmat că aceasta este cea mai interesantă problemă de informatică pe care au întâlnit-o vreodată. Punctajul mediu obținut a fost de 47.03 puncte.

Datorită multitudinii de cazuri care trebuie tratate era aproape imposibil ca cineva să rezolve problema pentru toate configurațiile posibile. Fiecare concurent a găsit diverse configurații pentru tabla de joc în funcție de anumite caracteristici ale dimensiunilor *m* și *n*.

În final, se pare că cel mai bine s-a descurcat *Mihai Stroe* din București care a obținut cel mai mare punctaj. *Virgil Șerbănuță* din București a obținut punctaje maxime pentru 18 din cele 20 de teste, dar celelalte două configurații indicate de el nu respectau condițiile date, motiv pentru care nu a obținut nici un punct la aceste teste.

Vă prezentăm în continuare implementarea propusă de *Adrian Cârcu* din Bistrița (90 puncte):

Listing JOC.PAS

```
{$S-,Q-,R-,G+,E-,N+,A+,B-}
const crlf:array[1..2] of Char=#13#10;
type ar=array[1..500] of Char;
var fis:Text;
    fis2:file;
    m,n,i,j,k:Integer;
    x:array[1..500] of ^ar;
    s:array[1..1002] of Char;
    swaped:Boolean;

procedure mod2mod2;
begin
  for i:=1 to m do
    for j:=1 to n do
      if (i=1)or(i=m) then
        if (j=1)or(j=n) then
          x[i]^[j]:='3'
        else x[i]^[j]:='5'
      else if (j=1)or(j=n) then
             x[i]^[j]:='5'
           else if (i mod 3=0) and
                                (j mod 3=0)
                then x[i]^[j]:='8'
                else x[i]^[j]:='7'
end;
```

CR3P3 Top 10

1.	Mihai Stroe, București	94
2.	Cristian Cadar, București	93
3.	Robert Baban, Baia Mare	92
3.	Silviu Bota, Baia Mare	92
5.	Adrian Cârcu, Bistrița	90
5.	Virgil Șerbănuță, București	90
7.	Gabriel Erzse, Oradea	85
8.	Daniel Dumitran, București	84
9.	Cosmin Negruseri, Bistrița	82
10.	Mihai Pătrașcu, Craiova	80

```
procedure mod2;
begin
  for i:=1 to m do
   for j:=1 to n do
     if i mod 3=0 then
       if (j=1)or(j=n) then
         x[i]^[j]:='4'
       else x[i]^[j]:='6'
     else if (j=1)or(j=n) then
            x[i]^[j]:='3'
          else x[i]^[j]:='5'
end;

procedure mod0mod0;
begin
  if m<n then
    begin
      i:=m; m:=n; n:=i; swaped:=true
    end;
  for i:=1 to m-1 do
    for j:=1 to n-2 do
      if (i mod 3=0) or (j mod 3=0)
        then x[i]^[j]:='4'
      else x[i]^[j]:='3';
  for i:=1 to n do x[m]^[i]:='2';
  x[m]^[n-1]:='0';
  for i:=1 to m-1 do
    if i mod 3=0 then
      begin
        x[i]^[n-1]:='6'; x[i]^[n]:='4'
      end
    else
      begin
        x[i]^[n-1]:='5'; x[i]^[n]:='3'
      end
end;
```

```
procedure mod0mod1;
begin
  for i:=1 to m do
    for j:=1 to n do
      if i=m then
        if (j=n-1)or(j=n-2) then
          x[i]^[j]:='0'
        else x[i]^[j]:='2'
      else if (j=n-1)or(j=n-2) then
             if i mod 3=0 then
               x[i]^[j]:='6'
             else x[i]^[j]:='5'
           else if (i mod 3=0) or
                                (j mod 3=0)
                then x[i]^[j]:='4'
                else x[i]^[j]:='3'
end;

procedure mod1mod1;
begin
  x[1]^[1]:='0';
  x[m]^[n]:='0';
  x[1]^[n]:='2';
  x[m]^[1]:='2';
  for j:=2 to n-1 do
    if j mod 3=1 then
      begin
        x[1]^[j]:='4'; x[m]^[j]:='4'
      end
    else
      begin
        x[1]^[j]:='3'; x[m]^[j]:='3'
      end;
  for i:=2 to m-1 do
    if i mod 3=1 then
      begin
        x[i]^[1]:='4'; x[i]^[n]:='4'
      end
    else
      begin
        x[i]^[1]:='3'; x[i]^[n]:='3'
      end;
  for i:=2 to m-1 do
    for j:=2 to n-1 do
      if ((i mod 3=2)and(j mod 3=0))or
         ((i mod 3=0)and(j mod 3=2))or
         ((i mod 3=1)and(j mod 3=1))
        then x[i]^[j]:='6'
        else x[i]^[j]:='5'
end;

procedure eq1;
begin
  if n=1 then
    begin
      i:=m; m:=n; n:=i; swaped:=true
    end;
  for i:=1 to n do
    if i mod 3=0 then x[1]^[i]:='2'
    else x[1]^[i]:='1';
  if n mod 3=0 then x[1]^[n]:='0';
  if n mod 3=1 then
  begin
    x[1]^[n]:='0'; x[1]^[n-1]:='0'
  end
end;

Begin
  Assign(fis,'joc.in'); Reset(fis);
  Read(fis,m,n); Close(fis);
  for i:=1 to 500 do New(x[i]);
  if (m=1)or(n=1) then eq1
  else
  begin
    if (m mod 3<>2)and(n mod 3=2) then
    begin
      i:=m; m:=n; n:=i; swaped:=true
    end;
    if (m mod 3=1)and(n mod 3=0) then
    begin
      i:=m; m:=n; n:=i; swaped:=true
    end;
    if (m mod 3=2)and(n mod 3=2) then
      mod2mod2
    else if m mod 3=2 then mod2
    else if (m mod 3=0)and(n mod 3=0)
           then mod0mod0
    else if (m mod 3=0)and(n mod 3=1)
           then mod0mod1
    else mod1mod1
  end;
  Assign(fis2,'JOC.OUT');
  Rewrite(fis2,1);
  Fillchar(s,Sizeof(s),' ');
  if swaped then
  begin s[m*2]:=#13; s[m*2+1]:=#10 end
  else
  begin s[n*2]:=#13; s[n*2+1]:=#10 end;
  if swaped then
    for i:=1 to n do
    begin
      for j:=1 to m do
        s[(j shl 1)-1]:=x[j]^[i];
      Blockwrite(fis2,s,m*2-1+2)
    end
  else
    for i:=1 to m do
    begin
      for j:=1 to n do
        s[(j shl 1)-1]:=x[i]^[j];
      Blockwrite(fis2,s,n*2-1+2)
    end;
  Close(fis2)
End.
```

Runda 5

CR5P1 Triunghi

Se dă următorul triunghi:

```
                1
              2 3
            4 5 6
          7 8 9 10
       11 12 13 14 15
     16 17 18 19 20 21
   22 23 24 25 26 27 28
 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45
. . . . . .
```

Să se găsească trei numere care formează un triunghi echilateral a căror sumă este X ($1 \le X \le 10^{18}$), astfel încât două numere se află pe aceeași linie, iar al treilea pe o linie cu indice mai mic.

Numărul X se citește din fișierul **TRIUNGHI.IN**. Cele trei numere vor fi scrise în fișierul **TRIUNGHI.OUT** în ordine crescătoare, pe o singură linie, separate printr-un singur spațiu. Dacă există mai multe soluții, va fi afișată numai una. Dacă nu există soluție, se va afișa 0 0 0 (trei de 0). Programul va trebui să se execute în cel mult o secundă.

Exemple

TRIUNGHI.IN
```
101
```

TRIUNGHI.OUT
```
15 41 45
```

TRIUNGHI.IN
```
20
```

TRIUNGHI.OUT
```
0 0 0
```

CR5P2 Beculețe

Fie o rețea ortogonală ale cărei linii sunt la distanță unitară, iar nodurile ei sunt în puncte de coordonate întregi din intervalul [-30000,30000]; pătratele rețelei sunt colorate inițial cu alb (culoarea de indice 0).

În n ($1 \le n \le 255$) noduri ale rețelei se află câte un beculeț. Unind în linie dreaptă, printr-un fir, două beculețe, toate pătratele traversate efectiv de fir (firul trece prin interiorul pătratului) își schimbă culoarea din culoarea i în culoarea i+1. Instalația funcționează dacă și numai dacă între oricare două becuri există legătură (directă sau indirectă).

Determinați o modalitate de conectare a celor n becuri astfel încât instalația să funcționeze, iar coloratura rețelei să fie minimă. Prin coloratura rețelei se înțelege suma indicilor de culoare a tuturor pătratelor din rețea.

Datele de intrare se citesc din fișierul **BECULETE.IN** care va conține pe prima linie numărul n al beculețelor, iar pe următoarele n linii coordonatele acestora.

Datele de ieșire vor fi scrise în fișierul **BECULETE.OUT** care va conține pe prima linie coloratura rețelei, pe a doua linie numărul m al legăturilor directe realizate, iar pe următoarele m linii câte 4 numere separate printr-un singur spațiu care reprezintă coordonatele a două beculețe unite printr-un fir direct. Programul va trebui să se execute în cel mult cinci secunde.

Exemplu

BECULETE.IN
```
5
0 0
2 10
1 5
5 7
10 0
```

BECULETE.OUT
```
12
4
2 10 5 7
1 5 0 0
5 7 1 5
10 0 0 0
```

CR5P3 Multiplu

Se dau numerele naturale n ($2 \le n \le 15000$) și b ($2 \le b \le 9$). Scrieți un program care determină, în mai puțin de o secundă, un multiplu al numărului n care este format numai din cifrele 0 și b.

Datele de intrare se citesc din fișierul **MULTIPLU.IN** care conține, pe o singură linie, numerele n și b, separate printr-un singur spațiu.

Datele de ieșire vor fi scrise în fișierul **MULTIPLU.OUT** care va conține multiplul determinat.

Exemple

MULTIPLU.IN
```
40 3
```

MULTIPLU.OUT
```
3000
```

MULTIPLU.IN
```
40 4
```

MULTIPLU.OUT
```
40
```

Introducere în teoria GRAFURILOR

Cristian Cadar

Arborii parțiali de cost minim reprezintă un domeniu de maximă importanță a teoriei grafurilor, cu numeroase aplicații practice în toate domeniile tehnice. De aceea, în acest articol vă prezentăm doi dintre cei mai cunoscuți algoritmi de aflare a unui arbore parțial de cost minim: algoritmul lui Kruskal și algoritmul lui Prim.

Înainte de abordarea acestui subiect, de mare importanță pentru numeroase domenii de activitate, vom familiariza cititorul cu câteva noțiuni legate de arbori, necesare înțelegerii acestui articol.

Definiție

Un graf neorientat, conex și aciclic se numește arbore.

Definiție

Fie G un graf. Numim arbore parțial al grafului G un graf parțial al lui G care este arbore.

Principalele proprietăți ale arborilor pot fi sintetizate prin următoarele afirmații echivalente:

- *G* este un arbore cu *n* noduri;
- *G* este un graf conex cu *n*-1 muchii;
- *G* este un graf aciclic cu *n*-1 muchii;
- *G* este un graf conex minimal (prin suprimarea unei muchii, graful obținut devine neconex);
- *G* este un graf aciclic maximal (prin adăugarea unei muchii, graful obținut va conține cicluri);
- oricare două vârfuri din *G* sunt unite printr-un lanț unic.

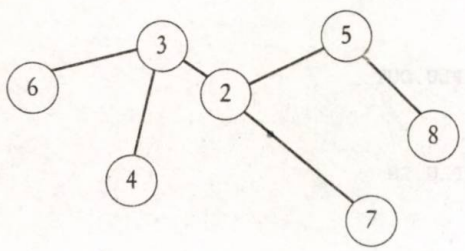


Fig. 1: Exemplu de arbore

Demonstrarea acestor proprietăți este foarte simplă și o lăsăm pe seama cititorului.

În majoritatea problemelor, arborii sunt folosiți pentru ierarhizarea informațiilor ce corespund vârfurilor arborelui. De aceea au fost introduși arborii cu rădăcină.

Definiție

Un arbore cu rădăcină este o mulțime finită A de noduri astfel încât:

- *există un nod numit rădăcină;*
- *toate celelalte noduri sunt repartizate în m mulțimi disjuncte $A_1, A_2, \ldots A_m$, fiecare mulțime fiind la rândul său arbore. Între rădăcina arborelui și rădăcinile arborilor $A_1, A_2, \ldots A_m$ există câte o muchie.*

Observăm că un arbore oarecare poate fi transformat într-un arbore cu rădăcină prin alegerea arbitrară a unui nod din arbore drept rădăcină.

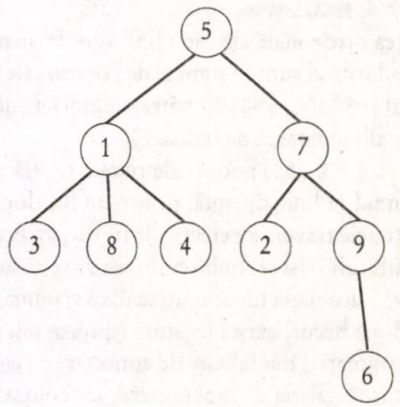


Fig. 2: Exemplu de arbore cu rădăcină

În ceea ce privește arborii, majoritatea termenilor folosiți aici derivă din vocabularul arborelui genealogic. Astfel afirmăm că rădăcina este *tatăl* rădăcinilor arborilor A_1, A_2, ..., A_m, iar acestea din urmă sunt *fii* ai rădăcinii arborelui. Nodurile cu același tată (părinte) sunt numite *frați*. În mod similar sunt folosiți termenii de *bunic*, *nepot*, *descendent*, *ascendent* etc.

Un arbore cu rădăcină poate fi etajat (ierarhizat). Considerăm rădăcina ca aflându-se pe nivelul 0, descendenții direcți ai acesteia (fiii) pe nivelul 1, descendenții direcți ai acestora din urmă pe nivelul 2 etc. De exemplu, pentru arborele din figura 2, nodul 5 se află pe nivelul 0, nodurile 1 și 7 pe nivelul 1, nodurile 3, 8, 4, 2 și 9 pe nivelul 2, iar nodul 6 pe nivelul 3.

Cu aceste notații, putem aborda subiectul acestui articol și anume *generarea arborilor parțiali de cost minim*. Arborii parțiali de cost minim (pe scurt A.P.M.) au numeroase aplicații în diverse domenii. Exemplul cel mai des folosit este unul din domeniul transporturilor: *dându-se n orașe ale unei țări, se cere interconectarea lor, astfel încât lungimea totală a șoselelor ce trebuie construite să fie minimă. Interconectarea orașelor presupune ca între oricare două orașe să existe cel puțin un drum (nu neapărat direct).*

Această problemă, ca de altfel multe altele de acest tip, poate fi modelată folosind teoria grafurilor. Structura de orașe poate fi codificată printr-un graf neorientat, ale cărui noduri reprezintă orașele, iar muchiile reprezintă șoselele care ar putea fi construite. Lungimile muchiilor sunt date bineînțeles de lungimile șoselelor corespunzătoare. Problema se reduce astfel la găsirea unui graf parțial conex, pentru care suma lungimilor muchiilor să fie minimă. Se observă cu ușurință faptul că pentru a fi conex, graful parțial nu trebuie să conțină decât n-1 muchii, deci avem de a face cu un arbore.

Există doi algoritmi foarte cunoscuți de determinare a unui A.P.M. al unui graf: algoritmul lui *Kruskal*, de complexitate $O(m \cdot \log n)$ și algoritmul lui *Prim*, de complexitate $O(n^2)$ (pentru acest algoritm se pot obține și complexități mai bune, așa cum va fi explicat ulterior). Ambii algoritmi folosesc tehnica *greedy*. Mai precis se pornește cu o mulțime de arbori A (inițial A conține doar arbori cu un singur nod), care reprezintă subarbori ai unui A.P.M. La fiecare pas al algoritmului, se adaugă o muchie w în A, astfel încât $A \cup \{w\}$ să constituie de asemenea un set de subarbori ai unui A.P.M. Astfel, după introducerea a n-1 muchii, mulțimea A va reține chiar un A.P.M. al grafului dat.

Deși tehnica de abordare este aceeași pentru ambii algoritmi, modul în care se realizează alegerea muchiilor diferă. Vom începe prin prezentarea algoritmului Kruskal, algoritm ce are o complexitate de $O(m \cdot \log n)$. Mulțimea A de mai sus este inițializată cu n arbori A_1, A_2,.... A_n, fiecare arbore A_i (i=1..n) conținând doar nodul i. La fiecare din cei n-1 pași, algoritmul unește doi subarbori din mulțimea A, rezultând în final un A.P.M. al grafului dat. Folosind o tehnică *greedy*, algoritmul Kruskal alege din toate muchiile ce unesc doi arbori din A, muchia de lungime minimă. Muchiile selectate nu trebuie memorate în cadrul mulțimii A, ele putând fi afișate la fiecare din cei n-1 pași ai algoritmului.

Un prim șablon al algoritmului Kruskal este următorul:

```
Kruskal (graf G)
APM={}
A1={1}, A2={2}, ..., An={n}
pentru pas = 1,n-1
   Alege muchia minimă (i,j) care unește doi
                                   arbori Ap și Aq
   APM=APM+{(i,j)}
   Unește arborii Ap și Aq
```

O primă optimizare a algoritmului de mai sus constă în a sorta inițial muchiile grafului, descrescător în funcție de lungimea acestora. Algoritmul devine:

```
Kruskal(graf G)
APM={}
A1={1}, A2={2}, ..., An={n}
Sortează descrescător muchiile grafului în funcție de cost
pentru k = 1,m
   dacă muchia k unește doi arbori Ap și Aq
        atunci APM=APM+{k}
               Unește arborii Ap și Aq
```

Observăm că mulțimile A_i sunt mulțimi de noduri. Condiția ca o muchie k să fie o muchie care să unească doi arbori A_p și A_q, este echivalentă cu condiția ca cele două extremități ale muchiei să nu se găsească într-un același arbore A_i. O primă variantă de memorare a arborilor A_i este aceea de a reține pentru fiecare nod i din graf, arborele căruia îi aparține. De exemplu, dacă pentru un graf cu 11 vârfuri avem arborii A_1={2, 5, 8}, A_2={1}, A_3={3, 9, 10}, A_4={4, 6, 7, 11}, atunci structura va fi memorată prin vectorul A=(2, 1, 3, 4, 1, 4, 4, 1, 3, 3, 4). Folosind această reprezentare, complexitatea algoritmului va fi $O(m \cdot n)$, deoarece determinarea submulțimii din care face parte un nod se face în $O(1)$, iar unirea a doi arbori în $O(n)$. Pentru a obține complexitatea dorită $O(m \cdot \log n)$, vom folosi o modalitate eficientă de memorare a acestor mulțimi disjuncte. Pentru aceasta, fiecare mulțime va fi structurată sub forma unui arbore cu rădăcină. Astfel, mulțimile din exemplul anterior ar putea fi memorate astfel:

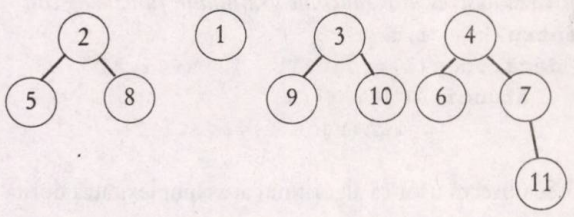


Fig. 3: Reprezentarea mulțimilor disjuncte

Pentru fiecare nod al grafului, reținem tatăl său. Prin definiție, un nod fără tată, are tatăl 0. Deci, în cazul de mai sus vectorul t al taților va fi: (0, 0, 0, 0, 2, 4, 4, 2, 3, 3, 7). Având această structură, unirea a doi arbori se face foarte ușor: tatăl rădăcinii unuia dintre arbori devine rădăcina celuilalt arbore:

```
union (nod a,b) // unește subarborii de
                // rădăcini a și b
t(a)=b
```

Presupunând că arborii sunt echilibrați (cel mai mare nivel atins în arbore este $\lceil \log_2 n \rceil$), rezultă că determinarea rădăcinii arborelui din care face parte un anumit nod se face în complexitatea $O(\log n)$:

```
root (nod a)
dacă t(a)=0
  atunci root=a
  altfel root=root(t(a))
```

Problema care apare constă în faptul că în urma operației union, arborele nou format s-ar putea să nu mai fie echilibrat. Pentru a ne asigura că vom forma tot un arbore echilibrat, tatăl rădăcinii unui arbore va reține în loc de valoarea 0, numărul nodurilor arborelui respectiv. Pentru a putea diferenția în continuare rădăcina de celelalte noduri ale arborelui, acest număr va fi reținut cu semn negativ. În cazul arborilor din figura 1, vectorul t va fi (-3, -1, -3, -4, 2, 4, 4, 2, 3, 3, 7). Unirea a doi arbori va putea fi făcută astfel încât în orice moment arborii A_i să fie echilibrați:

```
union (nod a,b)
k=t(a)+t(b)
dacă t(a)>t(b),
  atunci t(b)=a
         t(a)=k
  altfel t(a)=b
         t(b)=k
```

Având implementate aceste proceduri, algoritmul Kruskal devine:

```
Kruskal (graf G)
APM={}
t(1)=-1, t(2)=-1, ..., t(n)=-1
Sortează descrescător muchiile grafului în funcție de cost
pentru k = 1,m
  dacă root(i)≠root(j)   // k=(i,j)
    atunci APM=APM+{k}
           union(root(i),root(j))
```

Se observă ușor că algoritmul are complexitatea dorită $O(m \cdot \log n)$.

În cele ce urmează vom demonstra corectitudinea algoritmului Kruskal. În primul rând se observă ușor faptul că algoritmul generează un arbore parțial, deoarece el selectează n-1 muchii ale grafului care nu formează cicluri. Să arătăm că acest arbore parțial este și de cost minim. Fie $(a_1, a_2, ..., a_{n-1})$ arborele parțial generat de algoritmul Kruskal, unde $a_1 \le a_2 \le ... \le a_{n-1}$ sunt muchiile arborelui. Fie $B=(b_1, b_2, ..., b_{n-1})$ un A.P.M. al grafului respectiv, cu $b_1 \le b_2 \le ... \le b_{n-1}$. Fie k primul indice pentru care $a_k \neq b_k$. Bineînțeles $a_k < b_k$, altfel algoritmul Kruskal ar fi ales muchia b_k, fiind de cost mai mic și neformând cicluri cu muchiile $a_1, a_2, ..., a_{k-1}$. În continuare, adăugăm la A.P.M.-ul B muchia a_k. Se va obține graful $G'= (a_1, a_2, ..., a_{k-1}, b_k, ..., b_{n-1}, a_k)$, care va conține un ciclu generat de introducerea muchiei a_k. Cum a_k nu formează cicluri cu muchiile $a_1, a_2, ..., a_{k-1}$, înseamnă că aceasta va forma cicluri cu muchiile $b_k, ..., b_{n-1}$. Dar $a_k < b_k, ..., b_{n-1}$, deci eliminând din ciclul format o muchie diferită de a_k, vom obține un arbore parțial al cărui cost este mai mic decât cel al arborelui B. Contradicție cu faptul că B este A.P.M., deci nu există un indice k pentru care $a_k \neq b_k$.

Această demonstrație este deosebit de importantă, ea conferindu-ne o informație foarte utilă despre A.P.M.-urile unui graf. Ea arată că două A.P.M.-uri nu pot diferi prin lungimile muchiilor, ci doar prin muchiile în sine. O consecință imediată a acestui fapt este că pentru un graf în care oricare două muchii au lungimi diferite, există un A.P.M. unic. Aceste observații sunt foarte importante deoarece ne permit să aflăm cu ușurință toți A.P.M. ai unui graf.

În cele ce urmează, vom descrie algoritmul lui Prim. Vom insista mai puțin asupra acestui algoritm, deoarece el folosește ideea algoritmului *Dijkstra*, algoritm care a fost descris detaliat în cadrul episodului anterior. În acest caz mulțimea A este mulțimea S din cadrul algoritmului Dijkstra. La fiecare din cei n-1 pași, în S este introdus un nou nod, care se află la distanță minimă de nodurile deja introduse.

```
Prim(graf G, nod s)
Inițializare S, DIST                  // ca la Dijkstra
pentru pas=1,n-1
(1) selectează nodul v∉S pentru care
                                      DIST(v) este minim
(2) S=S+{v}
(3) pentru w∉S și există muchia (v,w)
      dacă DIST(w) > cost(v,w)
(4)     atunci DIST(w) = cost(v,w)
```

Analiza complexității acestui algoritm este identică cu cea făcută în cazul algoritmului Dijkstra. Implementând pe S sub formă de vector, vom avea complexitatea $O(n^2)$, implementându-l sub formă de heap, vom avea $O(m \cdot \log n)$, iar implementându-l sub formă de heap Fibonacci, vom avea $O(n \cdot \log n + m)$.

Cristian Cadar este student în anul II la Universitatea Politehnică București. Poate fi contactat prin e-mail la adresa ccadar@pcnet.pcnet.ro.

Pagini WEB cu JavaScript

Mihai Scorțaru

În cadrul acestui ultim episod al serialului despre limbajul JavaScript vom prezenta unul dintre cele mai importante elemente de care se ține seama în proiectarea paginilor Web și anume cookie-urile. Acestea sunt folosite pentru a permite utilizatorilor să navigheze mai ușor prin site-uri complexe.

Inițial cookie-urile erau folosite doar de script-uri aflate pe server, dar prin intermediul limbajului *JavaScript* ele sunt acum accesibile și pe mașina client. Termenul ***cookie*** se referă la orice obiect folosit pe Web pentru a stoca informații despre sistemul clientului. Toate aceste informații sunt păstrate într-un singur fișier **COOKIES.TXT** care conține câte un cookie pe fiecare linie. Transferul datelor păstrate într-un cookie este realizat prin intermediul antetelor HTTP (pachete folosite de server și de client pentru a se identifica). Deoarece există o mulțime de programe de navigare care nu oferă încă suport pentru cookie-uri ar trebui să vizitați din când în când pagina Web de la adresa `http://www.research.digital.com/nsl/formtest/stats-by-est/NetscapeCookie.html` pentru a consulta lista programelor de navigare care suportă cookie-uri.

Limbajul *JavaScript* oferă posibilitatea creării de cookie-uri din cadrul paginilor pe care le proiectați. Cookie-urile pot fi folosite în foarte multe moduri, dar cel mai important rol al acestora este de a păstra o anumită informație care să fie disponibilă pentru mai multe pagini Web. De exemplu, dacă pagina corespunzătoare fișierului **PAG1.HTML** conține un formular care este completat, în momentul vizitării paginii Web **PAG2.HTML** care conține un formular foarte asemănător, ar fi bine ca datele care au fost deja completate să nu trebuiască reintroduse.

În momentul în care încărcați o pagină Web, valorile variabilelor din pagina anterioară sunt pierdute. Pentru a păstra informațiile păstrate prin intermediul acestor variabile puteți crea un cookie care poate fi accesat din pagina dorită.

Caracteristici

Fiecare cookie este format din cel mult șase elemente: `name`, `value`, `expires`, `domain`, `path` și `secure`. Cu ajutorul acestor elemente sunt păstrate atât domeniul de valabilitate al cookie-ului cât și datele propriu-zise.

Elementul name

Fiecare cookie trebuie să aibă un nume care să permită accesarea lui. Acest nume poate fi orice șir de caractere care nu conține spații, virgule sau două puncte. Dacă trebuie neapărat ca numele să conțină asemenea caractere, puteți folosi o codificare a acestora. Cookie-urile pot fi denumite la fel ca și variabilele.

Elementul value

Fiecare cookie trebuie să aibă o anumită valoare care reprezintă informația stocată în cookie-ul respectiv. Valoarea poate fi orice șir de caractere.

Elementul expires

Fiecare cookie "expiră" la o anumită dată care poate fi indicată prin intermediul unui șir special care are formatul: `Wdy, DD-Mon-YY HH:MM:SS GMT`, elementele șirului având semnificația știută. Cookie-urile vor exista până la data respectivă și vor dispărea imediat ce ora sistemului va fi cea indicată de valoarea acestui element. Specificarea unei astfel de date este opțională; dacă nu este specificată data expirării, cookie-ul va dispărea în momentul în care se va termina sesiunea de lucru, adică atunci când programul de navigare nu mai are nici o fereastră deschisă.

Elementul domain

Nu este normal ca un cookie creat de o pagină Web de pe un anumit site să afecteze pagini Web de pe un alt site. Din acest motiv fiecare cookie va avea asociat un domeniu. Pentru domeniile de nivel superior cum ar fi `.com`, `.edu` etc. numele domeniului va trebui să conțină cel puțin de două ori caracterul punct (`'.'`). În cazul domeniilor de nivel inferior cum ar fi `.uk`, `.ca`, `.ro` etc. numele domeniului va trebui să conțină acest caracter de cel puțin trei ori. Implicit domeniul este setat ca fiind domeniul corespunzător site-ului de pe care a fost încărcată pagina respectivă.

Trebuie menționat faptul că, din motive de securitate, numai site-urile care fac parte dintr-un anumit domeniu pot crea cookie-uri care să fie valabile pentru domeniul respectiv.

Elementul path

Un cookie poate să fie semnificativ numai pentru un anumit nivel al unui site. De exemplu, dacă un site conține pagini distincte pentru vizitatori înregistrați și pentru vizitatori neînregistrați, atunci cookie-urile ar putea fi semnificative numai pentru o anumită categorie. Valoarea acestui element va conține calea spre paginile pentru care este valabil cookie-ul respectiv. Prezența acestui element este opțională și, dacă lipsește, calea considerată este cea a paginii care a creat cookie-ul respectiv.

Elementul secure

Acest element are o valoare booleană (`true` sau `false`), valoarea implicită fiind `false`. În cazul în care este setată valoarea `true` pentru acest element, cookie-ul respectiv va fi disponibil numai pentru servere pe care programul de navigare le consideră sigure.

Limitări

Există o limită pentru numărul de cookie-uri pe care le poate crea un site pentru un anumit client. Nu pot fi create decât maximum 20 de cookie-uri pentru un anumit server sau domeniu.

De asemenea, pot exista maximum 300 de cookie-uri în total; în cazul în care există deja 300 de cookie-uri și se încearcă crearea altuia, unul dintre cookie-urile create anterior va fi șters.

Informațiile referitoare la un cookie nu pot avea o dimensiune mai mare decât 4096 de bytes.

Crearea unui cookie

Limbajul *JavaScript* oferă mai multe funcții pentru crearea, citirea și ștergerea cookie-urilor. În continuare vom prezenta o funcție care poate fi folosită pentru a crea un cookie. Funcția va avea între doi și șase parametri corespunzători celor șase elemente ale unui cookie. Singurii parametri necesari sunt numele și valoarea corespunzătoare cookie-ului; restul parametrilor sunt opționali.

```
function BakeCookie(name,value){
  var argv=BakeCookie.arguments;
  var argc=BakeCookie.arguments.length;
  var expires=(argc>2)?argv[2]:null;
  var path=(argc>3)?argv[3]:null;
  var domain=(argc>4)?argv[4]:null;
  var secure=(argc>5)?argv[5]:false;
  document.cookie=name+"="+escape(value)+
                  ((expires==null)?"":
    (";expires="+expires.toGMTString()))+
      ((path==null)?"":(";path="+path))+
((domain==null)?"":(";domain="+domain))+
      ((secure==true)?"; secure" : "");}
```

Pentru crearea unui cookie vom putea folosi apelul:

```
BakeCookie("MyCookie","MyValue",
                        "","","",true);
```

Citirea unui cookie

În momentul în care se încearcă apelarea unui cookie, se caută în fișierul **`COOKIES.TXT`** numele cookie-ului respectiv. Dacă există mai multe cookie-uri cu acest nume, se alege cookie-ul care are valoarea corectă pentru elementul `path`. Este returnată valoarea cookie-ului corespunzător sau `null` dacă cookie-ul nu a fost găsit. Operațiile descrise anterior pot fi implementate astfel:

```
function EatCookie(name){
  var arg=name+"=";
  var alen=arg.length;
  var clen=document.cookie.length;
  var i=0;
  while(i<clen){
    var j=i+alen;
    if(document.cookie.substring(i,j)==arg)
      return EatCookieVal(j);
    i=document.cookie.indexOf(" ",i)+1;
    if(i==0) break;
  }
  return null;
}

function EatCookieVal(offset){
  var endstr=document.cookie.
                    indexOf(";",offset);
  if (endstr==-1)
  endstr=document.cookie.length;
  return unescape(document.cookie.
              substring(offset,endstr));
}
```

Ștergerea unui cookie

Operația de ștergere a unui cookie este destul de simplă; nu trebuie decât să setăm valoarea elementului `expires` pe `null` și cookie-ul va fi șters când se va închide programul de navigare. O altă variantă o reprezintă setarea unei date de expirare anterioare momentului curent. Modalitatea de ștergere a unui cookie este prezentată în cele ce urmează:

```
function TossCookie(name){
  var exp=new Date();
  exp.setTime(exp.getTime() - 1);
  var cval=GetCookie (name);
  document.cookie=name+"="+cval+
        ";expires="+exp.toGMTString();
}
```

Folosirea cookie-urilor

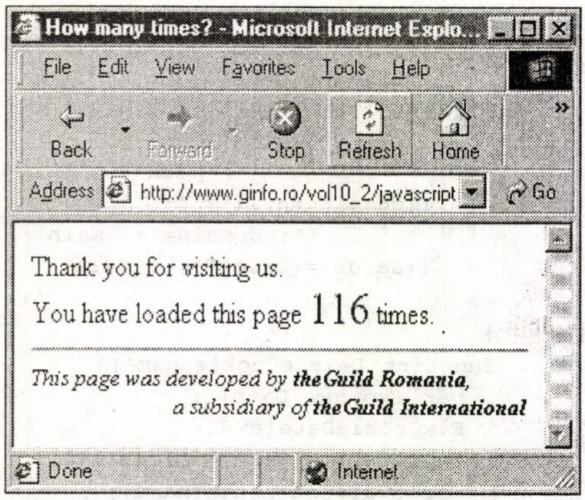
Posibilitatea stocării de informații despre un anumit utilizator fac din cookie-uri un instrument foarte puternic și folositor. În continuare vom prezenta două exemple de utilizare a cookie-urilor și anume *determinarea numărului de accesări ale paginii curente de către un anumit vizitator*, precum și *avertizarea vizitatorului asupra apariției unei modificări în pagină față de ultima vizită*.

Numărul de vizite

Documentul HTML corespunzător unei pagini Web care afișează de câte ori a fost accesată pagina respectivă de către un anumit vizitator este prezentat în continuare:

```
<HTML>
  <TITLE>
    How many times?
  </TITLE>
  <HEAD>
    <SCRIPT language="Javascript">
      function getCookieVal(offset){
        var endstr=document.cookie.
                    indexOf(";",offset);
        if (endstr==-1)
          endstr=document.cookie.length;
        return unescape(document.cookie.
              substring(offset,endstr));
      }
      function GetCookie(name){
        var arg=name+"=";
        var alen=arg.length;
        var clen=document.cookie.length;
        var i=0;
        while(i<clen){
          var j=i+alen;
          if (document.cookie.
                    substring(i,j)==arg)
            return getCookieVal(j);
          i=document.cookie.
                      indexOf(" ",i)+1;
          if (i==0) break;
        }
        return null;
      }
      function SetCookie(name,value){
        var argv=SetCookie.arguments;
        var argc=SetCookie.arguments.
                                  length;
        var expires=(argc>2)?
                          argv[2]:null;
        var path=(argc>3)?
                          argv[3]:null;
        var domain=(argc>4)?
                          argv[4]:null;
        var secure=(argc>5)?
                         argv[5]:false;
        document.cookie=name+"="+
                              escape(value)+
          ((expires==null)?"":
  ("; expires="+expires.toGMTString()))+
          ((path==null)?"":
                      ("; path="+path))+
          ((domain==null)?"":
                    ("; domain="+domain))+
          ((secure==true)?"; secure":
                                    "");
      }
      function DeleteCookie(name){
        var exp=new Date();
        FixCookieDate(exp);
        exp.setTime (exp.getTime()-1);
        var cval=GetCookie(name);
        if (cval!=null)
         document.cookie=name+"="+cval+
         "; expires="+exp.toGMTString();
      }
      var expdate = new Date();
      var num_visits;
      expdate.setTime(expdate.
          getTime()+(5*24*60*60*1000));
      if (!(num_visits=GetCookie
                        ("num_visits")))
        num_visits = 0;
      num_visits++;
      SetCookie("num_visits",
                    num_visits,expdate);
      document.write("Thank you for"+
                    "visiting us.<BR>");
      document.write("You have loaded"+
          "this page <FONT"+ " size=5>"+
         num_visits+"</FONT> times.<BR>");
    </SCRIPT>
  </HEAD>
  <BODY>
    <HR>
    <FONT size=2>
      <I>
      This page was developed by
      <B>theGuild Romania</B>,
      <BR>        
             
             
         
      a subsidiary of
      <B>theGuild International</B>
      </I>
    </FONT>
  </BODY>
</HTML>
```

La un moment dat pagina Web corespunzătoare documentului HTML anterior ar putea arăta astfel:



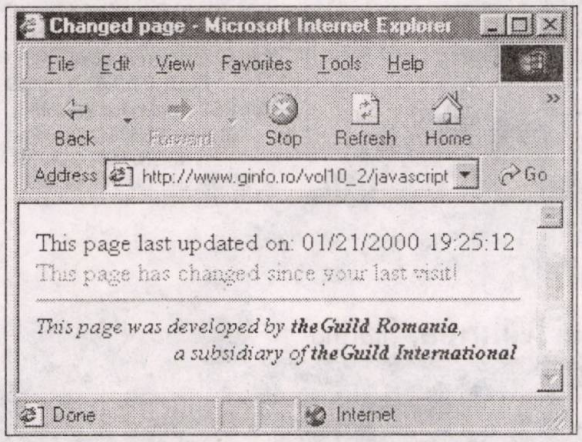
Pagină modificată

Chiar dacă cookie-ul prezentat anterior poate fi uneori folositor, puțini vizitatori sunt interesați de numărul de accesări pentru o anumită pagină. Pe de altă parte, un vizitator ar putea fi interesat dacă o anumită pagină s-a modificat de la ultima vizită. Următorul exemplu prezintă un document HTML care poate fi folosit în acest scop (nu este cea mai eficientă soluție, dar folosește cookie-uri):

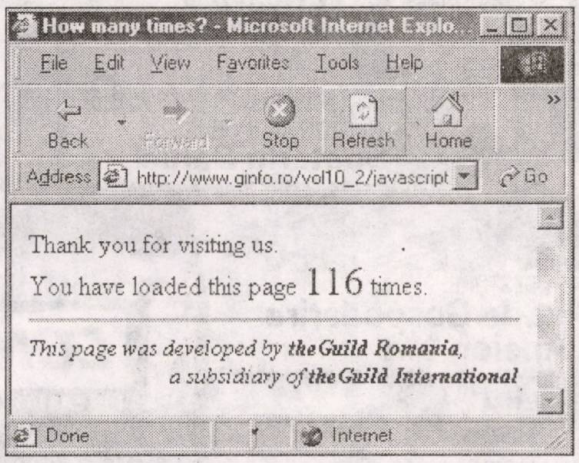
```
<HTML>
  <HEAD>
    <TITLE>
      Changed page
    </TITLE>
    <SCRIPT language="JavaScript">
      function getCookieVal(offset){
        var endstr=document.cookie.
                    indexOf (";",offset);
        if (endstr==-1)
          endstr=document.cookie.length;
        return unescape(document.cookie.
              substring(offset,endstr));
      }
      function GetCookie(name){
        var arg=name+"=";
        var alen=arg.length;
        var clen=document.cookie.length;
        var i=0;
        while(i<clen){
          var j = i + alen;
          if (document.cookie.substring
                                (i,j)==arg)
            return getCookieVal(j);
          i=document.cookie.
                      indexOf(" ",i)+1;
          if (i==0) break;
        }
        return null;
      }
      function SetCookie(name,value){
        var argv=SetCookie.arguments;
        var argc=SetCookie.arguments.
                                length;
        var expires=(argc>2)?
                          argv[2]:null;
        var path=(argc>3)?
                          argv[3]:null;
        var domain=(argc>4)?
                          argv[4]:null;
        var secure=(argc>5)?
                          argv[5]:false;
        document.cookie=name+"="+
                          escape(value)+
        ((expires==null)?"":
                ("; expires="+expires.
                      toGMTString()))+
        ((path==null)?"":
                      ("; path="+path))+
        ((domain==null)?"":
                ("; domain="+domain))+
        ((secure==true)?"; secure":"");
      }
      function DeleteCookie(name){
        var exp=new Date();
        FixCookieDate(exp);
        exp.setTime(exp.getTime()-1);
        var cval=GetCookie(name);
        if (cval!=null)
          document.cookie=name+"="+
                cval+"; expires="+exp.
                          toGMTString();
      }
      var cookie_date=new Date
              (document.lastModified);
      var expdate = new Date();
      expdate.setTime(expdate.
          getTime()+(5*24*60*60*1000));
      document.write("This page last"+
                        "updated on: "+
                document.lastModified);
      document.write("<BR>");
      if (!(cookie_date ==
            GetCookie("cookie_date"))){
        SetCookie("cookie_date",
                  cookie_date,expdate);
        document.write("<FONT color="+
          "'Red'>This page has changed"+
              " since your last visit!"
                      +"</FONT><BR>");
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <HR>
    <FONT size=2><I>
```

```
        This page was developed by
        <B>theGuild Romania</B>,
        <BR>        
               
               
           
        a subsidiary of <B>theGuild
        International</B>
        </I>
    </FONT>
  </BODY>
</HTML>
```

În cazul în care pagina este vizitată fără ca ea să fi fost modificată de la ultima vizită, fereastra programului de navigare ar putea arăta astfel:



În cazul în care pagina a fost modificată în intervalul de timp scurs de la ultima vizită, sau în cazul în care aceasta este prima vizită a paginii, fereastra programului de navigare ar putea arăta astfel:



La fel ca și în cazul exemplului anterior, se verifică existența cookie-ului, iar în cazul în care acesta nu este găsit, se creează un nou cookie. Valoarea corespunzătoare cookie-ului este data și ora ultimei modificări a documentului respectiv. Testul constă în verificarea datei și orei corespunzătoare cookie-ului și a celei corespunzătoare ultimei modificări a documentului curent. În cazul în care cele două momente coincid, pagina nu a mai fost modificată; în caz contrar pagina a fost modificată și în fereastra programului de navigare apare un mesaj care informează vizitatorul asupra acestui fapt.

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro

News...

XHTML 1.0

Consorțiul World Wide Web (W3) recomandă utilizarea noilor specificații XHTML (eXtensible HTML), pentru a mări aria de utilizare a tehnologiei XML (eXtensible Markup Language), fără a face inutilizabile elemetele HTML actuale. XHTML 1.0 a fost creat prin rescrierea HTML 4.0 într-un format XML, creându-se astfel specificații care vor putea fi folosite cu browsere HTML, dar vor profita și de avantajul oferit de tehnologia XML. Conform declarațiilor oficiale ale Consorțiului W3, XHTML va avea rolul unui pod de legătură între un Internet bazat în mare măsură pe HTML și beneficiile adoptării tehnologiei XML. Dezvoltatorii care folosesc HTML 4.0 nu ar trebui să aibă mari probleme în legătură cu trecerea la noile specificații. Instrumentele necesare convertirii documentelor HTML 4.0 în XHTML 1.0 vor fi oferite de Consorțiul W3. Odată cu adoptarea noilor specificatii, dezvoltatorii nu vor mai avea probleme cu rescrierea documentelor pentru diferitele tipuri de echipamente.

Puteți afla mai multe despre noile specificații XHTML 1.0 în următoarele episoade ale serialului *Pagini WEB cu...*

NORTON CLEAN SWEEP
Pitiș Cătălin, Brașov

NORTON UTILITIES
S.C. TEROM, Iași

NORTON ANTI VIRUS
Spătariu Daniel, Iași

EASY RESTORE
Sist.de Gospodărire a Apelor, Deva

PARTITION MAGIC
Vânzariuc Ioan, Botoșani

LOST & FOUND
SPIMC, Ploiești

ROMSYM DATA-SRL
SOFTWARE

COREL DRAW 8
Constantin Marius, Galați

OMNILOGIC BGS

PROLINK
Chiș Felician, Tășnad - SM

ALPIS

CALCULATOR ALPIS
Dumitrescu Nic. Florian, Dr.Turnu Severin

BRINEL

Microsoft Windows NT Windows 98

WINDOWS 2000 READY PCs

ALPIS®

ISO-9002 BRINEL PC PROD

COMPAQ

P/N : 131015-a82
Deskpro EN SFF C466
6.4 GB Hard
64 MB RAM
NIC / Sound / W95
Monitor 15"

MAINBOARD
QDI WinneX 1, Intel 82810,
Intel Celeron 400PPGA W/128K cache
32 MB SDRAM DIMM PC 100 (8ns)
SVGA OB BOARD 4MB AGP
HDD 6.4 Quantum Fireball
CD-ROM 40x IDE
SOUND CARD on board +
Stereo Speakers 60W PMPO
FLOPPY 3.5", 1.44MB
ACER PS2 Keyboard WIN'95 +
Mouse SERIAL 3b + Pad
MONITOR COLOR
SVGA SMILE 14" DIGITAL

CALCULATOR COMPAQ
Răileanu Andrei Mihnea, Slatina

MONITOR 17"
Chirilă Ciprian Dan, Iași

SPRINT computers

MONITOR SPRINT 17" - model MP 1770N

Fii un PC Player adevărat!

PixelView PlayTVpro

PV-BT878P+(W/FM)
Watch TV +Play TV+ • Motion Capture+
Teletext+ • Closed Capture+
FM Radio + • Remote Control+
Video Camera+ • Video Conferencing Ready

oferit de

AGER
THE COMPUTER COMPANY

Anul 2000 se lasă cu PREMII

COMPUTER PRESS AGORA

Microsoft® România

MICROSOFT OFFICE 2000 - ROMANIAN
Dumitru Marius, Fulga de Jos - Prahova

Micr Office 2000 Professional

TORNADO SISTEMS

ZIP DRIVE IOMEGA
Baicu Ciprian Nicolae, Dr.Turnu Severin

Iomega Zip Drive
250 MB

Canon

Canon Bubble Jet
Color Pri

CALCULATOR VOG + IMPRIMANTA CANON
Berechet Luminița, București

Procesor: AMD-K6 400 MHz 3DNow!
64 MB RAM • HD 6,7 Gb
Video: S3 Savage3D 8 MB Monitor 17"
Sound card pe placa de bază: EESS TM ES1938S Solo-1
CD-ROM: LITEON LTN382 40X
Tastatură: Qtronix multimedia Scorpius-98N
Mouse: Qtronix Lynx 3D
Modem: HSF 56k date/fax/voce intern, cu cipset Rockwell
Boxe: Mecmar LS-26A, 135W PMPO

GT Grup Transilvae

SCANNER UMAX
Casa de Asigurări de Sănătate, Cluj-N

UMAX Astra 1220U
rezoluție optică 600x1200dpi
max. 9600x9600, col. 36bit, 1 pass

CALCULATOR SISTEC
Pop Adrian, Baia-Mare

HEWLETT PACKARD
ROMÂNIA

IMPRIMANTA HP
Bors Istvan, Tg.Mureș

HP DeskJet 610 C

...za ATX, INTEL 810, W/4 MB AGP&SOUND, UDMA 66
Procesor INTEL CELERON 400 (128kc) PPGA • Memorie 32 MB SDRAM...
Video 4 MB AGP ON BOARD • Hard-disk QUANTUM 6.4 GB...
Tastatură GENIUS WIN 98 • Floppy 1.44 MB FDD...
Monitor 14" SVGA LR NI DIGITAL MPRII • CD...
Sound ON BOARD AC 97 AUDIO.
Boxe 120 W, ACTIVE SP...

S-AU ABONAT și au câștigat!

Abonează-te
la PCMagazine România,
PC Report sau Gazeta de informatică,
reviste editate de Computer Press Agora, până la data de 15 decembrie 1999.
Vei participa la tragerea la sorți care va desemna câștigătorii premiilor.
Trimite cartile poștale din aceste reviste pe adresa:
Computer Press Agora, str.T.Vladimirescu nr.63/1, Tg.Mureș 4300 (CP 230/1)
Pentru informații suplimentare poți suna la: tel. 065-166516, fax 065-166290

ALGORITMI...

la gura sobei

Sorin Groza

Iarna este unul dintre anotimpurile cele mai îndrăgite de copii, atât pentru jocurile de iarnă dar mai ales pentru faptul că vine Moș Crăciun. Nici în această perioadă algoritmii nu au fost uitați. Mai ales că afară este frig, iar la gura sobei este locul ideal pentru a "coace" niște algoritmi. În așteptarea lui Moș Crăciun, dar și auzind că pentru anumiți algoritmi care să-l ajute în muncă, Moșul va oferi frumoase cadouri, vă prezentăm câteva probleme compuse și rezolvate împreună cu Ionuț Stan, un elev pasionat de informatică.

Itinerarul lui Moș Crăciun

Cu ocazia sărbătorilor de iarnă Moș Crăciun a primit scrisori de la foarte mulți copii. Deoarece este ocupat cu procurarea și repartizarea cadourilor după dorința fiecărui copil, Moșul îl roagă pe spiridușul informatician să scrie un program care va stabili traseul lui Moș Crăciun. Traseul va fi o listă care va conține orașele împreună cu numărul de copii din fiecare oraș, în ordinea descrescătoare a numărului de copii, iar în caz de egalitate a numărului de copii din două sau mai multe orașe, acestea vor fi afișate în ordine alfabetică.

Intrare

```
n = 6
Aiud, Vaslui, Aiud, Deva, Deva, Arad
```

Ieșire

```
Itinerarul lui Mos Craciun:
Aiud   2
Deva   2
Arad   1
Vaslui 1
```

Soluție

După citirea orașelor se va ordona alfabetic șirul acestora, deoarece la număr egal de copii afișarea se va face alfabetic. Se construiește șirul `f` care conține numărul de copii din fiecare oraș. Se ordonează șirul orașelor (folosind vectorul `f`) în ordinea descrescătoare a numărului de copii. După aceea se va trece la afișarea listei, utilizând șirul orașelor (ordonat descrescător după numărul de copii, iar în caz de egalitate în ordine alfabetică) și șirul `f`.

Listing ITINERAR.PAS

```
Program itinerar;
var n,i,j,k,a:Word;
    aux,xua:string;
    x:array[1..100] of string[100];
    f:array[1..100] of Integer;
    gata,termin:Boolean;

Begin
  Write('Cati copii au trimis scrisori');
  Write(' lui Mos Craciun ? ');
  Readln(n);
  for i:=1 to n do
  begin
    Write('Orasul copilului ',i,' - ');
    Readln(x[i])
  end;
  { ordonarea alfabetica a oraselor }
  repeat
    gata:=true;
    for i:=1 to n-1 do
      if x[i]>x[i+1] then
      begin
        aux:=x[i];
        x[i]:=x[i+1];
        x[i+1]:=aux;
        gata:=false
      end
  until gata;          { sfarsit ordonare }
    { determinarea numarului de copii din }
    {                        fiecare oras }
```

```
  for i:=1 to n do
    for j:=1 to n do
      if x[i]=x[j] then Inc(f[j]);
        { ordonarea sirurilor oraselor si }
        {  a numarului de copii din orase }
  repeat
    termin:=true;
    for i:=1 to n-1 do
      if f[i]<f[i+1] then
      begin
        a:=f[i]; f[i]:=f[i+1]; f[i+1]:=a;
        xua:=x[i]; x[i]:=x[i+1];
        x[i+1]:=xua; termin:=false
      end
  until termin;          { sfarsit ordonare }
  Writeln('Itinerarul lui Mos Craciun :');
  for i:=1 to n do
    if x[i]<>x[i+1] then
      Writeln(x[i],' ',f[i])
End.
```

O problemă pentru Moș Crăciun și nu numai...

În drum spre copiii care i-au trimis scrisori, Moșul trece și pe lângă geamul casei a doi copii săraci. Văzând bradul frumos împodobit s-a gândit că și cei doi copii ar merita câte un cadou. El are în sanie n saci, iar în fiecare sac un număr de cadouri. Deoarece este așteptat de alți copii de la care a primit scrisori, Moșul nu are timp să desfacă sacii și hotărăște să lase doi saci. Pentru a duce cât mai multe cadouri la ceilalți copii, el trebuie să aleagă doi saci astfel încât numărul cadourilor rămase în ceilalți saci să fie cât mai mare și să se împartă exact la cei n copii care îl așteaptă. Care vor fi sacii pe care îi va lăsa Moșul și câte cadouri va primi fiecare copil? În situația în care Moșul nu poate să facă o astfel de alegere, vă lăsam pe voi să hotărâți cum se va descurca Moșul.

Intrare

```
n=6
8,3,12,9,10,7
```

Ieșire

```
Mosul da sacii 2 si 5
Fiecare copil primeste 6 cadouri
```

Soluție

În variabila s vom calcula suma cadourilor din saci, deoarece pentru a afla numărul de cadouri, mai puțin doi saci, vom scădea din numărul total de cadouri numărul cadourilor din cei doi saci (variabila d). Vom încerca pe rând toate combinațiile de doi saci și, de fiecare dată, vom vedea dacă respectă condiția cerută. Dacă vom găsi o soluție mai bună, vom păstra în variabilele sac1 și sac2 indicii sacilor. În final vom tipări aceste valori, precum și numărul cadourilor primite de fiecare copil.

Listing SAC.PAS

```
Program sac;
var n,i,j,sac1,sac2,d,s,max:Inteer;
    x:array[1..1000] of Integer;
Begin
  Write('Cati saci are in sanie Mosul ');
  Readln(n);
  Writeln('Dati numarul de cadori din');
  Write(' fiecare sac');
  for i:=1 to n do read(x[i]);
  for i:=1 to n do s:=s+x[i];
        { s = numarul de cadouri din saci }
  for i:=1 to n-1 do
    for j:=i+1 to n do
    begin
      d:=x[i]+x[j];
    { d = nr. de cadouri din sacii i si j }
      if ((s-d) mod n=0) and (s-d>max) then
      begin
        max:=s-d; sac1:=i; sac2:=j
      end
    end;
  Writeln('Mosul da sacii ',sac1)
  Write(' si ',sac2);
  Writeln('Fiecare copil primeste ')
  Write(max div n,' cadouri')
End.
```

Moș Crăciun are pretenții!

După cum bine se cunoaște, una dintre dorințele lui Moș Crăciun la întâlnirea cu un copil este aceea de a-l întreba care au fost notele primite în timpul semestrului. Deoarece timpul lui Moș Crăciun este foarte limitat (datorită faptului că trebuie să meargă în foarte multe locuri) el a lansat următoarea idee. Copiii nu vor citi toate notele din carnet, ci vor proceda astfel: mai multe note consecutive din carnet, egale între ele, vor fi înlocuite de cuvintele "x de y", unde x reprezintă numărul de note egale, iar y valoarea acestora. Este clar că timpul în care un copil își spune notele va fi mai scurt. Scrieți un program care afișează lista cerută de Moș.

Intrare

```
n=9
9,9,9,9,10,10,10,8,8
```

Ieșire

```
Lista pentru Mos Craciun:
4 de 9
3 de 10
2 de 8
```

Soluție

Pentru a transcrie șirul de note conform cerinței problemei, vom incrementa un contor (k) cât timp avem elemente consecutive egale. Când se termină o astfel de

secvență vom tipări "k de x[i]". Vom parcurge șirul de note de la început și până la sfârșit, astfel că, în final, vom obține lista pentru Moș Crăciun.

Listing PRETENTII.PAS

```
Program pretentii;
var i,k,j,n:Integer;
    x:array[1..100] of Integer;
Begin
  Write('Care este numarul de note ');
  Readln(n);
  Writeln('Introduceti notele din carnet :');
  for j:=1 to n do
    Read(x[j]);
  Writeln('Lista pentru Mos Craciun:');
  i:=1;
  while (i<n) or (i=n) do
  begin
    k:=1;
          { aflarea numarului de elemente}
          {           consecutive egale }
    while x[i]=x[i+1] do
    begin
      k:=k+1;
      i:=i+1
    end;
    Writeln(k,' de ',x[i]);
    { sfarsit aflare si tiparire secventa }
    i:=i+1
  end
End.
```

Vă propunem să rezolvați și voi câteva probleme la "gura sobei" și să i le prezentați lui Moș Crăciun la următoarea vizită.

Sărbătoare în lanț

În ultima perioadă de timp au fost o mulțime de sărbători, unele mai lungi, altele mai scurte. Se citește numărul de zile (ale sărbătorii) și o literă reprezentând codificarea sărbătorii respective. O secvență de cel puțin două zile codificate la fel reprezintă o sărbătoare în lanț. Să se determine numărul sărbătorilor în lanț și cea mai lungă dintre ele. Pe un rând se va afișa un număr, reprezentând numărul sărbătorilor în lanț, iar pe celălalt rând se va scrie codul sărbătorii și începutul, respectiv sfârșitul acesteia.

Intrare

```
n = 12
N,N,L,C,C,C,C,L,S,R,R,L
```

Ieșire

```
3
C 4 7
```

Gospodinele

În cartier s-a zvonit că se va scumpi untul. Toate gospodinele au dat fuga la magazinul alimentar pentru cumpărături de sărbători. Fiecare gospodină va cumpăra unt de toți banii de care dispune, la prețul indicat de vânzătoare. În urma cumpărării, fiecare gospodină va rămâne cu o sumă de bani. Să se determine suma de bani cu care au rămas cele mai multe gospodine. Gospodinele care nu au bani nici măcar pentru un pachet de unt, nu vor fi luate în considerare. Se vor citi de la intrare: numărul de gospodine, suma de bani a fiecărei gospodine, apoi prețul unui pachet de unt.

Intrare

```
5      - număr gospodine
22, 2, 28, 20, 9
3      - preț pachet unt
```

Ieșire

```
1
```

Parcarea

În oraș s-a construit o parcare foarte frumoasă. Pentru cei care administrează această parcare, ea se consideră a fi descrisă printr-un dreptunghi. O mașină parcată este "păstrată" în calculatorul firmei printr-o pereche (i,j). În această parcare există și o porțiune de parcare pentru VIP-uri, în care bineînțeles nu poate parca oricine. Dacă, totuși, o persoană neautorizată va parca aici, atunci aceasta va fi amendată. Citindu-se de la intrare numărul de mașini, pozițiile lor și cele două colțuri opuse ale porțiunii interzise, scrieți un program care va afișa numărul amenzilor aplicate.

Intrare

```
7          - numărul mașinilor
(2,3); (3,2); (5,3); (6,1); (5,4); (6,5);
(4,6) - poziții mașini
(6,2) - (3,5)      - colțuri parcare VIP
```

Ieșire

```
4 amenzi
```

Câștigătorii din luna ianuarie a premiilor
MSDN Universal Subscription
oferite de către Microsoft Romania sunt:

Airinei Ana
Str. Morii nr. 266, 5773 - Rădășeni, jud. Suceava

Alexandrescu Cristian
Cal. Națională, nr. 70, bl. G4, sc. B, et. 4, ap. 12
6800 - Botoșani

Bardac Mircea Ionut
O.P. 76 C.P. 144 București, tel. 01/7692438

Terminarea CURATĂ a unui program

Horea Haitonic

Un aspect deseori neglijat în realizarea unei aplicații cu caracter practic este generarea unei proceduri de "curățare" după terminarea programului. O astfel de procedură este foarte utilă în cazul programelor care folosesc diverse moduri grafice, în cazul programelor rezidente sau care redirectează întreruperi DOS.

În *Turbo Pascal*, "curățarea" se poate realiza folosind variabila `Pointer` predeclarată `ExitProc`. Aceasta indică adresa funcției care va fi apelată la terminarea programului, și poate fi modificată în vederea inserării unei funcții de terminare definită de programator după următorul procedeu:

```
OldExit:=ExitProc;   Salvez vechea valoare a pointerului ExitProc
ExitProc:=MyExit;    ExitProc indică acum procedura MyExit
```

unde `OldExit` este de tip `Pointer` iar `MyExit` este noua procedură de "curățare".

Procedura `MyExit` (fără argumente, compilată după modelul ***far*** cu directiva `$F+`) va fi în acest caz adăugată unei stive a procedurilor de terminare (aceste proceduri vor fi apelate pe rând după principiul *Last In First Out*), și are "datoria" de a reface vechea valoare a pointer-ului `ExitProc` (valoare salvată în prealabil în `OldExit`) pentru ca lanțul procedurilor de terminare a programului să nu fie întrerupt (evident fiecare funcție adăugată stivei va face același lucru).

Exemplu

Să presupunem că am realizat un program care redirectează întreruperea DOS `$1C` (*User Timer Tick*) spre procedura **`interrupt`** `NewProc` astfel:

```
GetIntVec($1C,oldvec);
SetIntVec($1C,NewProc);
```

În acest caz, pentru a ne asigura că la terminarea programului (indiferent în ce condiții) vechea întrerupere va fi refăcută, procedura `MyExit` poate fi de forma:

```
{$F+}
procedure MyExit;
begin
  SetIntVec($1C,oldvec);
  ExitProc:=OldExit
end;
{$F-}
```

Observație

Compilarea cu modelul de memorie ***far*** este absolut necesară pentru a permite așa numitele *far-jumps* (salturi mai lungi de 64 KB).

Procedura `MyExit` reface vechea întrerupere `$1C`, după care restaurează vechea valoare a pointerului `ExitProc` (pentru a nu întrerupe "lanțul", după cum s-a precizat mai sus).

O astfel de procedură de "curățare" poate afla cauza terminării programului folosind variabilele `ExitCode (Integer)` și `ErrorAddr (Pointer)`:

- La o terminare normală, `ExitCode` este `0`, iar `ErrorAddr` este **`nil`**.
- La terminare prin apelul procedurii `Halt`, valoarea variabilei `ExitCode` este cea a argumentului acestei proceduri, iar valoarea variabilei `ErrorAddr` este **`nil`**.
- La o terminare anormală, datorată unei erori, `ExitCode` conține codul erorii, iar `ErrorAddr` adresa instrucțiunii care a generat eroarea.

Astfel, putem adăuga procedurii `MyExit` următoarele linii:

```
Write('Terminare program:');
Write('Eroare nr. ',ExitCode);
```

```
Write(' la adresa ',seg(ErrorAddr^),':');
Writeln(ofs(ErrorAddr^));
```

pentru a afișa la terminarea (normală sau nu) a programului condițiile de ieșire.

În C++, implementarea unei proceduri de terminare este mai ușoară (și oarecum mai puțin puternică): se folosește același principiu al "stivei de funcții", dar introducerea unei funcții și menținerea continuității lanțului sunt gestionate de rutine interne. De remarcat însă că funcțiile de terminare nu se apelează decât în cazul ieșirii normale din program (lucru destul de normal de altfel, deoarece în C/C++ erorile nu au ca și consecință terminarea imediată a programului: funcția apelată întoarce o valoare din care se poate deduce eroarea apărută, iar execuția programului continuă). Folosirea unei funcții de "curățare" nu mai este în acest caz imperios necesară, dar simplifică munca programatorului, care nu trebuie să ia în calcul toate modalitățile de ieșire din program.

Se folosește funcția `int atexit(atexit_t func)` din `<stdlib.h>`, care adaugă `*func` la vârful stivei de funcții, scutind programatorul de efortul de a gestiona "lanțul". Tipul `atexit_t` este definit în `<stdlib.h>` astfel: `typedef void (*atexit_t)(void)`.

Reluăm exemplul prezentat anterior (întreruperea `$1C` va fi redirectată astfel:

```
#define __CPPARGS ...
void interrupt (*OldVect)(__CPPARGS);
OldVect=getvect(0x1C);
setvect(0x1C, NewHandler);
```

unde `void interrupt NewHandler(__CPPARGS)` este funcția care va fi asociată întreruperii).

Pentru a determina cauza ieșirii din program, se folosește variabila `errno` (`<stdlib.h>`) care conține numărul corespunzător erorii. Mai mult, funcția `void perror(const char *s)` din `<stdio.h>` poate fi folosită pentru a afișa numărul și semnificația ultimei erori apărute.

În acest caz, funcția `MyExit()` va fi introdusă în stivă prin `atexit(MyExit)` și va arăta astfel:

```
void MyExit(void){
  printf("Terminare program...");
  perror("");
  getch();
  setvect(0x1C,OldVect);
}
```

În ambele cazuri (*Turbo Pascal* și C++), la baza stivei de proceduri/funcții de terminare (și deci apelată ultima) se află cea definită în mod implicit de către bibliotecile de execuție, care face "curățarea" finală (închide fișierele `Input` și `Output`, respectiv `stdin` și `stdout`, afișează eventualele mesaje de eroare etc.)

Deși nu are un caracter vital, folosirea unor proceduri definite de utilizator, pentru terminarea corectă a programului în orice împrejurări, este utilă mai ales în cazul unui program "serios", care va fi utilizat de mai multe persoane și trebuie deci protejat împotriva eventualelor erori generate de aceștia. Mai mult, erori apărute în programele ce lucrează la nivel jos pot duce la blocarea sistemului, sau pot avea alte efecte extrem de neplăcute, în lipsa unor astfel de funcții de terminare. Din acest motiv, prezentarea făcută mai sus poate fi folositoare multor programatori care ar trebui să țină cont și de acest aspect al realizării unei aplicații.

Horea Haitonic este elev la liceul George Coșbuc din Năsăud; poate fi contactat la adresa horea_h@hotmail.com.

News...

Fuziune Corel/Inprise

Compania Corel a anunțat că fuzionează cu compania Inprise, printr-un schimb de acțiuni evaluat la 2,4 miliarde de dolari. Oficialii companiei au declarat că această fuziune va crea un pol de putere Linux. Mișcarea vine într-un moment în care este evident că ambele companii își concentrează atenția asupra sistemului de operare Linux. Până acum, Corel a dezvoltat aplicații Java, aplicații destinate echipamentelor mobile și pachete pentru birou, dar de curând a anunțat o inițiativă Linux majoră. În luna noiembrie a anului trecut, compania a anunțat o versiune desktop a sistemului de operare Linux, plănuind să lanseze în curând și o versiune Linux a pachetelor Office, CorelDraw și PhotoPaint. De cealaltă parte, Inprise a anunțat planuri de a oferi medii de dezvoltare Java pentru Linux și a declarat că pachetul Kylix, un set de instrumente rapide pentru dezvoltarea aplicațiilor Linux, va fi lansat în cursul acestui an.

Amenințarea GSM

Mihai Scorțaru

Este accesoriul sfârșitului de secol! Trebuie neapărat să îl aveți! Conferă poziție și putere, oferă posibilitatea comunicării cu restul lumii aproape din orice poziție a globului! Dar dacă va produce în creierul dumneavoastră o tumoare de dimensiunea unei mingi de ping-pong?

Peste 750.000 de români folosesc serviciile furnizorilor de telefonie mobilă și, probabil, în cursul anului 2000 cifra va depăși 1.000.000, adică aproape 10% din populația activă a țării. În ciuda dezvoltării spectaculoase a acestei modalități de comunicare, foarte puțini producători s-au gândit la posibilele efecte negative care pot apărea în cazul utilizării frecvente a telefoanelor mobile. În ultimul timp există dovezi din ce în ce mai concludente care arată că folosirea telefoanelor mobile este riscantă. Dl. Ross Adey, profesor de biochimie la Universitatea California din Riverside afirmă că "pentru prima dată în istorie ținem aproape de creier un emițător de mare putere".

Reacții

Când vorbiți la telefonul dumneavoastră mobil, vocea vă este transmisă prin intermediul antenei cu o radiație de frecvență radio (*RFR - Radio Frequency Radiation*) care variază între 450 și 1990 MHz, frecvență situate în domeniul microundelor. Nu este deloc surprinzător faptul că expunerea la astfel de microunde poate avea consecințe serioase asupra sănătății dumneavoastră.

În 1993, în Florida a existat un proces intentat de soțul unei femei care a murit din cauza unei tumori pe creier; soțul susținea că tumoarea a fost cauzată de folosirea frecventă a telefonului mobil. Chiar dacă nu a reușit să aducă dovezi în acest sens, procesul nu a rămas fără urmări; *CTIA* (*Cellular Telecommunications Industry Association*) a format un grup de cercetare numit *WTR* (*Wireless Technology Research*) care avea la dispoziție un buget de peste 25.000.000$ pentru a studia riscurile folosirii telefoanelor mobile.

După șase ani de studiu au apărut primele concluzii preliminare ale celor de la WTR; conducătorul acestui grup, dl. George Carlo, a declarat: "Am găsit dovezi de alterare genetică a sângelui uman. Există o mortalitate excesivă cauzată de cancer la creier printre utilizatorii de telefoane fără fir și există dovezi statistice clare a creșterii semnificative riscului apariției unei tumori neuroepiteliale. În acest moment avem mai multe date care sugerează probleme cu telefoanele mobile decât au avut cei de la FDA când au interzis implanturile cu silicon."

De ce nu reacționează industria telefoanelor mobile? Cercetările WTR au fost finanțate de către CTIA care colaborează intens cu producătorii de telefoane mobile și de infrastructură pentru comunicații fără fir și CTIA nu a încurajat publicarea rezultatelor cercetărilor.

Motorola, membru CTIA, și al doilea producător de telefoane mobile din lume a finanțat multe programe de cercetare a posibilelor efecte negative asupra sănătății asociate cu folosirea telefoanelor mobile; rezultatele acestor cercetări au fost rezumate de dl. Norman Sandler, director cu problemele strategice globale, prin cuvintele: "Produsele care respectă unele restricții stabilite nu expun sănătatea utilizatorilor la nici un risc cunoscut". Compania poate pune la dispoziție kilometri de hârtie care dovedesc fără echivoc absența oricărui risc pentru sănătatea utilizatorilor de telefoane Motorola.

Și totuși...

Dovezile aduse de Motorola ar putea să-i liniștească pe utilizatorii de telefoane mobile, dar se pare că lucrurile

nu stau chiar așa. Dl. Jerry Phillips a lucrat împreună cu Ross Adey la un proiect finanțat de Motorola la Universitatea California din Riverside începând cu 1991. Dânsul afirmă: "Relația [cu Motorola] a fost plăcută în primii ani până când am început să obținem primele date." Lucrurile s-au schimbat în momentul în care a fost stabilită o legătură între expunerea la RFR și apariția de tumori de creier la cobaii din laboratoare. Jerry Phillips a adăugat că Motorola este neînduplecată în cererea ei ca profesorul Adey să nu menționeze legătura dintre alterarea ADN-ului și expunerea la radiații de frecvență radio.

Profesorul Adey susține că "Motorola a manipulat rezultatele cercetărilor pe care eu și alții le-am raportat. De fapt ne-au tăiat fondurile deoarece eram prea curioși."; Norman Sandler, reprezentantul Motorola, neagă această afirmație. Este interesant faptul că unul dintre studiile lui Ross Adey a demonstrat faptul că există anumite frecvențe RFR pentru care frecvența tumorilor la creier în rândul cobailor a scăzut. Ne-am fi așteptat ca aceasta să fie o veste extraordinară pentru Motorola - imaginați-vă o campanie publicitară care să prezinte beneficiile folosirii telefoanelor mobile. Totuși Ross Adey a afirmat că Motorola nu a vrut să recunoască rezultatele care au indicat efecte biologice ale expunerii la RFR, fie ele pozitive sau negative.

Jerry Phillips, Ross Adey și alți membrii ai grupului susțin că văd o paralelă puternică între ceea ce se întâmplă acum și decadele în care producătorii de țigări au negat posibilitatea existenței unei legături între consumul de tutun și apariția cancerului pulmonar, în ciuda înmulțirii dovezilor științifice care demonstrau acest lucru. D-na Jo-Anne Basile, responsabilă cu relații externe și cu industria la CTIA nu este de acord cu grupul profesorului Adey; ea afirmă că "Nici nu se putea face o comparație mai inexactă. Industria nu a încercat niciodată să ascundă nici o informație sau comentariu negativ."

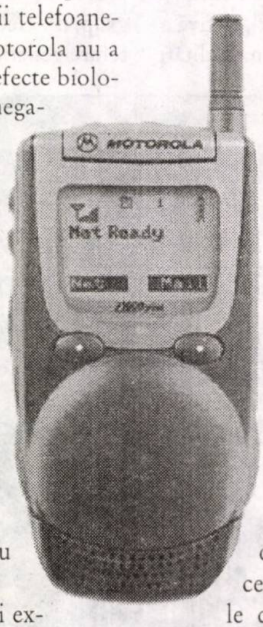
Nu este încă clară legătura dintre cancer și expunerea la RFR; când vorbiți la telefonul dumneavoastră mobil, 40% dintre radiațiile emise sunt absorbite de mână și cap, dar, de obicei, nivelul energetic nu este suficient de ridicat pentru a cauza apariția unei energii semnificative în interiorul capului. Deci care este cauza problemelor? "Aceasta este marea întrebare", a răspuns Jerry Phillips, "Dacă aș ști, acum aș da mâna cu regele Suediei în loc să-mi fac griji pentru finanțarea cercetărilor".

Există multe dovezi care indică alterarea ADN-ului ca fiind principalul "vinovat". Ross Adey a descoperit o legătură între microundele de intensitate scăzută și alterarea ADN-ului în creierul cobailor. Jerry Phillips a sugerat că, de fapt, RFR nu pot altera ADN-ul dar, într-un anume fel, pot stânjeni abilitatea de refacere a acestuia când este alterat prin cauze naturale. El afirmă că "ceea ce trebuie să știm este dacă există o interacțiune directă între RFR și ADN".

Jerry Phillips, Ross Adey și George Carlo afirmă că ei nu au misiunea de a falimenta giganții telecomunicațiilor. De fapt, toți trei posedă telefoane celulare și le folosesc ori de câte ori au nevoie fără a abuza însă. Ei afirmă că ar dori doar ca industria telecomunicațiilor să le accepte descoperirile și să permită publicarea acestora, iar consumatorii obișnuiți să hotărască singuri cum să reacționeze. În opinia lui George Carlo "ceea ce este greșit, este ascunderea informațiilor față de publicul larg". Atât companiile din industria telecomunicațiilor cât și oamenii de știință sunt de acord că sunt necesare cercetări suplimentare. Un studiu major, condus de Organizația Mondială a Sănătății, investighează efectele expunerii la RFR dar rezultatele cercetărilor sunt așteptate doar în anul 2005.

Concluzii

Între timp există un mod destul de simplu și nu foarte costisitor de a scăpa de expunerea la RFR în timpul folosirii telefonului mobil și anume folosirea unui headset: o cască de dimensiuni foarte mici în ureche și un microfon prins la reverul hainei permit poziționarea telefonului la o distanță sigură față de capul utilizatorului. Și acesta este cel mai bun loc posibil până când juriul deliberează...



Dacă nu, parafrazându-l pe Gordon Bass, redactor la revista *PC Computing* din Statele Unite, în loc de "Pentru limba română apăsați 1. For English press 2..." putem spune "Pentru cancer apăsați C. Dial C for cancer." Ar trebui să ne mulțumească cei de la DIALOG pentru reclama mascată pentru că se pare că "Viitorul sună bine!" pentru telefonia mobilă din România - ca să ne luăm revanșa față de cei de la CONNEX. Și în ultima vreme a apărut și SunTel...

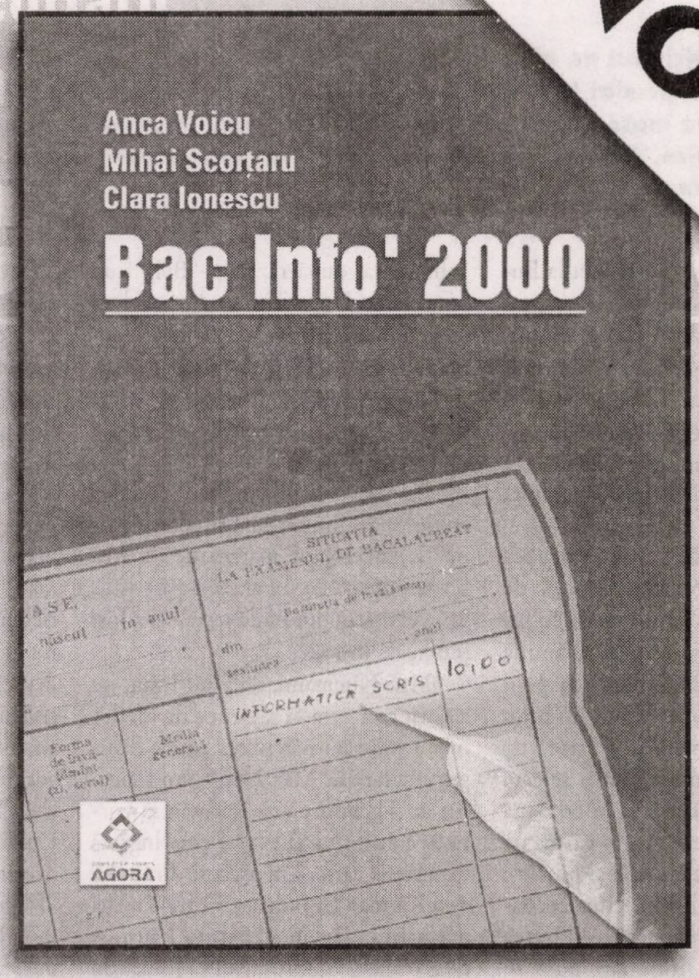
Dacă ar mai scădea prețurile ar fi excelent (bineînțeles și cele de la ROMTELECOM); dacă în vest companiile de telefonie mobilă încearcă să-și reducă tarifele pentru a fi comparabile cu cele pentru telefonia fixă, la noi singura companie de telefonie fixă își mărește tarifele până când devin comparabile cu cele pentru telefonia mobilă; sau le-au depășit deja? Cam așa se pare...

Se spune că tinerii români nu au acces la Internet; nici nu este de mirare pentru că puține familii își permit să plătească o factură telefonică de ordinul sutelor de mii de lei pentru ca să poată folosi cum se cuvine Internet-ul. În Statele Unite, chiar dacă familiile de acolo și-ar putea permite suportarea unei astfel de facturi, acest lucru nu este necesar, deoarece convorbirile locale sunt gratuite... Nu putem decât să sperăm că, în viitorul apropiat, vom ajunge și noi în aceeași situație.

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro

COMPUTER LIBRIS AGORA

NOUTATE



În această cărticică veți găsi subiecte de Bac la informatică și nu doar atât.
În partea a doua, vă punem la dispoziție posibile răspunsuri, soluții ale problemelor. Deoarece limbajul de programare este la alegere, depinde de candidat pe care îl preferă. Acolo unde răspunsul depinde de limbajul ales, veți găsi soluțiile corespunzătoare prezentate și în Pascal și în C.
La sfârșitul cărții sunt patru "variante" de subiecte propuse ale căror rezolvări le veți putea citi și studia în Gazeta de Informatică.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

AGORA

John von NEUMANN

Crina Groşan

Matematicianul de origine maghiară, John von Neumann, a emigrat în Statele Unite şi a predat la Universitatea din Princeton. A avut contribuţii majore în domeniile matematicii cuantice, în teoria jocurilor, teoria mulţimilor şi informatică. Arhitectura calculatoarelor actuale poartă numele de arhitectură von Neumann.

John von Neumann s-a născut la Budapesta; când era copil i se spunea Jancsi, diminutivul lui János, iar mai târziu, în Statele Unite, i s-a spus Johnny.

Tatăl lui John era bancher. Erau o familie de evrei în care practicarea credinţelor creştină şi iudaică nu mira pe nimeni.

Încă din copilărie, John dovedea o incredibilă memorie. La vârsta de 6 ani era capabil să glumească cu tatăl său în greaca clasică. Familia Neumann îşi delecta uneori oaspeţii cu demonstraţii ale abilităţii lui John de a memora numere din cartea de telefon. Unul dintre musafiri selecta la întâmplare o pagină şi o coloană de pe pagina respectivă din cartea de telefoane cu nume, adrese şi numere de telefon, tânărul John citea coloana, înapoia cartea oaspetelui şi putea răspunde oricărei întrebări puse în legătură cu datele respective sau să recite numerele şi adresele în ordine.

În 1911, John intră la Gimnaziul Lutheran. Şcoala avea o puternică tradiţie academică, ceea ce părea să conteze pentru tânărul John mai mult decât afinitatea religioasă. Profesorul său de matematică a recunoscut repede geniul lui Neumann şi l-a luat sub protecţia sa. Şcoala mai dăduse un alt matematician de marcă, cu un an înaintea lui Neumann şi anume pe Eugene Wigner.

Primul război mondial nu prea a avut influenţe asupra educaţiei lui John. Însă, după război, - Béla Kun controlând Ungaria timp de cinci luni cu un guvern comunist - familia Neumann a fost nevoită să plece în Austria datorită presiunilor la care a fost supusă. Dar, după o lună s-au reîntors la Budapesta.

În 1921, Neumann îşi termină studiile la Gimnaziul Lutheran. Prima sa lucrare în matematică a fost scrisă împreună cu Fekete - asistent la Universitatea din Budapesta - şi a fost publicată în 1922.

Max Nuemann, tatăl lui John, nu a dorit ca acesta să se ocupe de un domeniu care nu i-ar putea aduce câştiguri materiale. Max Neumann i-a cerut lui Theodore von Kármán să vorbească cu fiul său şi să-l convingă să urmeze o carieră în afaceri. Probabil von Kármán l-a îndrumat spre o astfel de carieră, dar, în sfârşit, toţi au fost de acord cu compromisul ca John von Neumann să urmeze studii universitare de chimie.

În Ungaria nu era o situaţie favorabilă pentru evrei; iată, de aici mai multe motive pentru care numărul studenţilor evrei care puteau intra la Universitatea din Budapesta era limitat. Desigur, chiar şi cu aceste restricţii, John von Neumann a reuşit uşor să câştige un loc în 1921 şi să studieze matematica. El însă, nu a participat la cursuri. În acelaşi an (1921) a intrat şi la Universitatea din Berlin ca să studieze chimia. A urmat cursurile Facultăţii de Chimie din Berlin până în 1923 când a plecat la Zürich.

A obţinut rezultate remarcabile la examenele de matematică de la Universitatea din Budapesta, în ciuda faptului că nu a participat la cursuri (de fapt nu participase la nici un curs). Von Neumann şi-a luat diploma în inginerie chimică la *Technische Hachschule* din Zürich, în 1926. Pe când se afla în Zürich nu şi-a pierdut interesul faţă de matematică (în ciuda studierii chimiei) şi a colaborat cu Weyl şi Pólya care erau amândoi în Zürich. El a ţinut chiar un curs al lui Weyl când acesta a lipsit din Zürich pentru o perioadă de timp. Colaboratorul său, Pólya, avea să declare mai târziu referitor la John: "John a fost singurul student de care mi-a fost vreodată frică. Dacă în timpul cursului lăsam o problemă nerezolvată, el venea la mine imediat ce se termina cursul cu soluţia completă pe un petec de hârtie".

Von Neumann şi-a luat doctoratul în matematică la Universitatea din Budapesta încă din 1926 cu o teză despre teoria mulţimilor. El publicase deja o definiţie a numerelor ordinale încă pe când avea 20 de ani, definiţia dată atunci fiind una folosită şi astăzi. Von Neumann a ţinut cursuri la Berlin între 1926-1929 şi la Hamburg între 1929-1930. A câştigat o bursă Rockfeller pentru a-şi completa studiile postdoctorale la Universitatea din Göttingen între 1926-1927. În

acest timp, von Neumann și-a câștigat statutul de celebritate. Și astfel, la 20 de ani, faima lui s-a răspândit în întreaga lume a matematicii. În 1929, Veblen l-a invitat pe von Neumann la Princeton pentru a ține discursuri despre teoria cuantică. Răspunzându-i lui Veblen că ar fi vrut să-și rezolve câteva chestiuni personale înainte de a pleca spre S.U.A, von Neumann a plecat la Budapesta unde s-a căsătorit cu Marietta Kövesi.

În 1930 devine profesor invitat la Princeton și va fi profesor aici până în 1933. Neumann a conferențiat așadar la Princeton, dar acesta nu era unul din punctele lui forte. Se spunea că era dificilă urmărirea lui de către cei care aleseseră să participe la cursuri; Neumann era notoriu în scrierea unor ecuații importante pe un petec liber de tablă pentru ca apoi să îl șteargă înainte ca studenții să le poată copia.

A devenit, în 1933, unul dintre primii șase profesori de matematică (J. W. Alexander, A. Einstein, M. Morse, O. Veblen, J. Von Neumann și H. Weyl) la noul institut de studii avansate din Princeton, poziție pe care și-a păstrat-o până la sfârșitul vieții.

În timpul primilor ani petrecuți în Statele Unite, von Neumann a revenit în Europa în perioada verii. Până în 1933 el și-a păstrat un post academic în Germania, dar a renunțat când naziștii au ajuns la putere. Neumann nu a fost refugiat politic; el venise în Statele Unite mai degrabă pentru a-și întări poziția academică pe care o câștigase în Germania.

Tot în 1933, devine co-editor al Analelor de Matematică și doi ani mai târziu co-editor al Compositio Mathematica. A rămas în aceste două funcții până la sfârșitul vieții sale.

Von Neumann și Marietta au avut o fiică, Marina, în 1936, dar căsătoria lor s-a sfârșit printr-un divorț în 1937.

În anul următor, von Nuemann s-a căsătorit cu Klára Dán, de asemenea din Budapesta. După căsătorie au revenit în Statele Unite. Aici von Neumann a dus o viață mai puțin obișnuită pentru un matematician de frunte. Îi plăceau foarte mult petrecerile. Una dintre cunoștințele sale spunea: "Petrecerile și viața de noapte au constituit o atracție specială pentru von Neumann. Încă pe când preda în Germania, von Neumann era un obișnuit al Cabaretelor din Berlin." Căsătorit acum cu Klára, petrecerile au continuat. Se spunea că petrecerile de acasă de la Neumann erau frecvente, faimoase și lungi.

Ulam, care a studiat creația lui Neumann, a afirmat că acesta era preocupat în munca sa din tinerețe nu numai de matematică, logică și teoria axiomatică a mulțimilor, ci simultan și de conținutul teoriei mulțimilor în general, obținând rezultate interesante în teoria măsurii și în teoria funcțiilor reale. Tot în perioada tinereții și-a început munca sa clasică la teoria cuantică și la noua mecanică statistică.

Lucrarea sa *Mathematische Grundlagen der Quantenmechanic* (1932) a constituit o parte importantă pentru noua mecanică cuantică. Van Hove scria "Mecanica cuantică a fost foarte norocoasă, dacă a reușit să atragă în primii săi ani după descoperire în 1925, interesul unui matematician de geniu de valoarea lui von Neumann. Ca rezultat, suportul matematic al acestei teorii a fost dezvoltat și aspectele formale ale tuturor noilor reguli au fost realizate de un singur om în doi ani (1927-1929)".

În teoria jocurilor, von Neumann a demonstrat teorema de minimax. Treptat, și-a extins munca în teoria jocurilor și în colaborare cu Oskar Morgenstern a scris *Theory of Games and Economic Behaviour*.

În 1938, ***Societatea Americană de Matematică*** l-a premiat pe von Neumann cu premiul Bôcher pentru lucrarea sa *Almost Periodic Functions and Groups*. Acesta a fost publicat în două părți în *Transactions of the American Mathematical Society*, prima parte în 1934, iar partea a doua un an mai târziu.

Von Neumann a fost unul din pionierii informaticii, având contribuții semnificative la dezvoltarea arhitecturii logice. El a avansat teoria locației de memorie, adoptând bit-ul ca unitate de măsură pentru memoria calculatoarelor și rezolvând problema comunicării între componentele calculatorului.

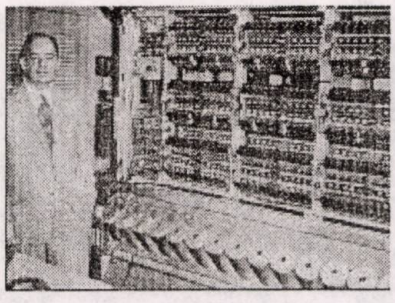
În timpul celui de al doilea război mondial a servit pe post de consultant al forțelor armate. În această calitate a avut contribuții la realizarea bombei atomice și a celei cu hidrogen.

Din 1940 a fost membru al Comitetului Științific de Avizare al Laboratorului de Cercetare Balistică din Maryland.

În 1955, președintele Statelor Unite l-a nominalizat în ***Comisia de Energie Atomică*** și în 1956 a primit premiul Enrico Fermi. Din păcate, în această perioadă, boala incurabilă de care suferea (cancer) s-a agravat.

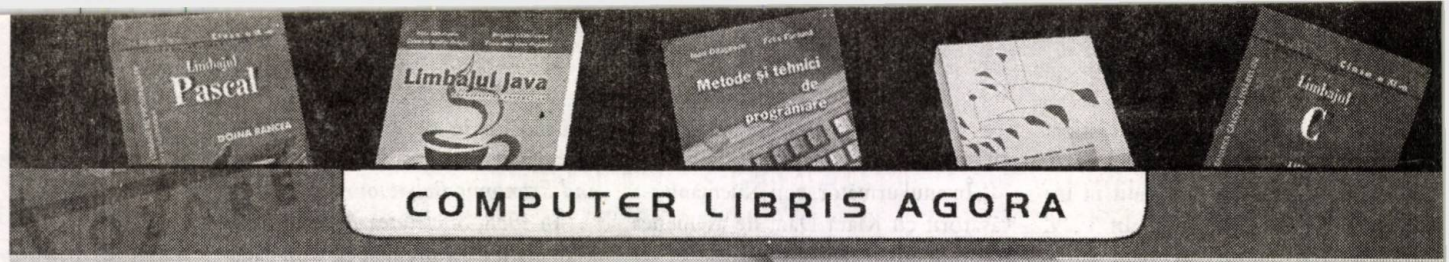
A murit un an mai târziu, în 1957. Numărul de onoruri și distincții care i-au fost acordate de universități și instituții de prestigiu din întreaga lume este impresionant.

Crina Groșan este studentă în anul V la Universitatea Babeș-Bolyai din Cluj și poate fi contactată prin intermediul redacției.



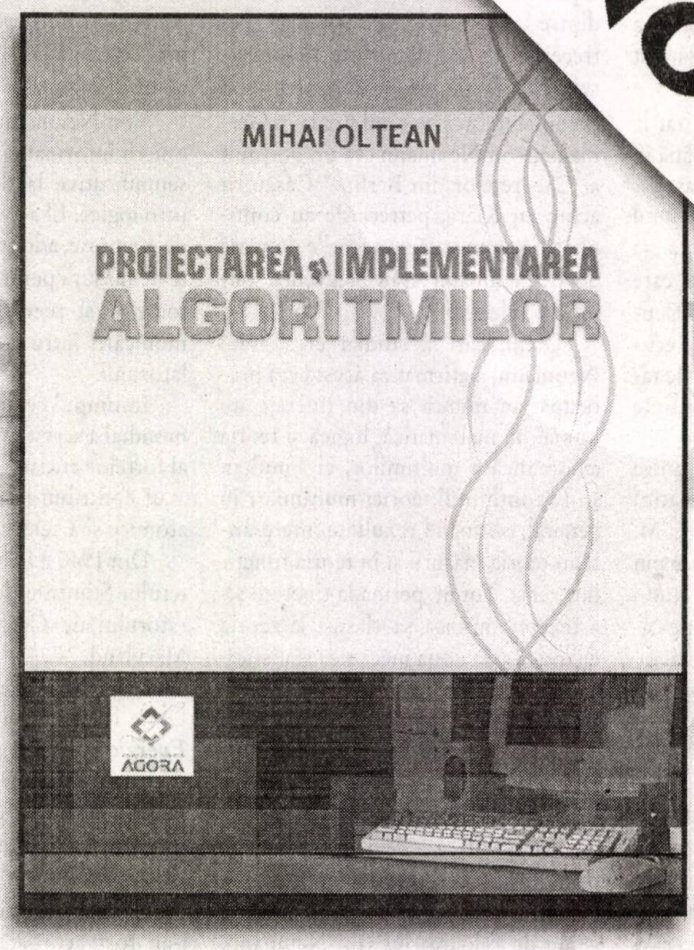
"If people do not believe that Mathematics is simple, it is only because they do not realize how complicated life is."

John von Neumann



COMPUTER LIBRIS AGORA

NOUTATE



Culegerea constituie un instrumentar pentru cei care doresc să intre în poseșia secretelor proiectării algoritmilor. Va fi util când vă pregătiți pentru teză, pentru examenul de bacalaureat sau pentru admitere. Veți găsi în culegere și probleme care pot fi analizate și rezolvate în cercurile de elevi și care vor fi utile în pregătirea pentru concursurile școlare.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Principiul optimalității

PROGRAMAREA dinamică în problemele NP-COMPLETE

Mihai Oltean

Pentru a aplica metoda programării dinamice este necesar să putem descompune problema inițială într-o serie de subprobleme, incluse unele în altele. Pornim apoi, cu calculul, de la subproblemele cele mai simple (care nu au nici o subproblemă în subordine) către subproblemele cele mai complexe (care se descompun direct, sau indirect) în tot mai multe subprobleme. În majoritatea cazurilor întâlnite, numărul subproblemelor este polinomial. Numărul subproblemelor în care se descompune problema inițială este de data aceasta exponențial.

Drum hamiltonian

Un comis-voiajor trebuie să viziteze o singură dată fiecare din cele $n+1$ orașe, plecând dintr-un oraș oarecare și întorcându-se tot în acel oraș. Între oricare două orașe există un drum direct care are atașat un cost numit distanță. Care drum minimizează distanța totală parcursă de comis-voiajor?

Rezolvare

Numerotăm orașele cu numere distincte din intervalul $[0, n]$.

Fără a pierde din generalitate, deoarece fiecare tur este ciclic, putem fixa originea în orașul 0. Presupunem că la un anumit pas al unui ciclu optim, care începe cu orașul 0, s-a ajuns la orașul i și au mai rămas k orașe $j_1, j_2, ..., j_k$ nevizitate înainte de a ne întoarce în 0. Este clar că, deoarece turul este optimal, acel drum care pornește din orașul i, trece prin orașele $j_1, j_2, ..., j_k$ și se întoarce în 0, trebuie să fie optim (să aibă lungime minimă), deoarece în caz contrar întregul ciclu nu ar mai fi optim, deoarece s-ar putea alege o cale mai scurtă de la i astfel încât să treacă prin orașele $j_1, j_2, ..., j_k$, iar apoi să revină în 0.

Să definim deci:

$c[i, j_1, j_2, ..., j_k]$ = lungimea unui drum minim între orașul i și orașul 0 care trece o singură dată prin fiecare din orașele nevizitate $j_1, j_2, ..., j_k$.

Soluția problemei se va afla în $c[0, j_1, j_2, ..., j_N]$.

Să notăm cu $d[i,j]$ distanțele directe dintre două orașe i și j. Atunci, ca o consecință a observației anterioare avem:

$$c[i, j_1, j_2, ..., j_k] = \min_{1 \le m \le k} \{d[i, j_m] + c[j_m, j_1, j_2, ..., j_{m-1}, j_{m+1}, ..., j_k]\}$$

Această formulă este o aplicație directă a principiului optimalității.

Pentru calcularea acestei formule începem cu:

$c[i, j] = d[i, j] + d[j, 0]$,

iar apoi calculăm pe $c[i, j_1, j_2]$, apoi pe $c[i, j_1, j_2, j_3]$ etc., iar în final pe $c[0, j_1, j_2, ..., j_N]$.

Secvența de valori j_m care minimizează expresia de mai sus, conduce la drumul minim.

Să vedem cum implementăm aceste formule într-o complexitate cât mai redusă. Cum se vede și din formula optimalității, numărul de parametri ai funcției c este mare (în cel mai defavorabil caz $N+1$). Dar, deoarece numerele $j_1, j_2, ..., j_k$ nu trebuie stocate în ordine, pentru reținerea lor nu este nevoie de un șir de lungime k, ci de o mulțime având cardinalitatea k. Se știe că mulțimile în *Turbo Pascal* sunt reprezentate la nivel de bit, deci în loc să avem o reprezentare a numerelor $j_1, j_2, ..., j_k$ pe k octeți, vom avea nevoie de doar $[k/8]+1$ octeți.

Nici codificarea sub formă de mulțimi nu este foarte economă din punct de vedere al memoriei ocupate. Orașele sunt etichetate cu numere întregi din intervalul $[0, n]$. La pasul unu (inițial) trebuie să calculăm numerele $c[i, j_1]$ care sunt în număr de N. La pasul 2 trebuie să calculăm numerele $c[i, j_1, j_2]$ care sunt în număr de C_N^k, ..., la un pas k trebuie să calculăm numerele $c[i, j_1, j_2, ..., j_k]$ care sunt în număr de C_N^k pentru un i fixat. Submulțimile $j_1, j_2, ..., j_k$ ale unei mulțimi de N elemente sunt în număr de C_N^k. Acestea se pot genera în ordine lexicografică (ca și în cartea de telefon). Mai mult decât atât, dându-se o submulțime având cardinalitatea k a unei mulțimi de cardinal N, se poate obține direct numărul ei de ordine lexicografică. De exemplu, submulțimea $\{1, 2, ..., k\}$ are numărul de ordine

lexicografică 1, iar submulțimea $\{N-k+1, ..., N\}$ are numărul de ordine lexicografică C_N^k. Procedeul este valabil și invers, adică, dându-se un număr de ordine lexicografică se poate genera direct (printr-un algoritm de complexitate $O(k)$) submulțimea corespunzătoare de cardinalitate k.

Deci, revenind la problema noastră, nu este nevoie de reținerea întregii submulțimi ci doar de un singur număr care să indice numărul de ordine lexicografică. Cel mai mare număr de ordine lexicografică C_N^k este acela pentru care k este jumătate din N.

Pentru a putea rezolva problema pentru un număr cât mai mare de orașe, stocarea acestei matrice se va face în *Heap*, adică accesul se va face în genul $c[i]^[j]$. Deoarece în momentul în care stocăm funcția $c[i, j_1, j_2, ..., j_k]$, trebuie să reținem și cardinalul mulțimii $\{j_1, j_2, ..., j_k\}$; la impementare vom avea de a face cu încă un parametru: lungimea combinării. Deci matricea în care vom construi soluția are trei parametri $c[i, k]^[w]$, unde i are semnificația de mai sus, k este lungimea combinării, iar w este numărul de ordine lexicografică al combinării $(j_1, j_2, ..., j_k)$ în cadrul mulțimii combinărilor de n elemente luate câte k.

Complexitatea va fi în acest caz de ordinul $O(N^3 2^N)$, dacă facem abstracție de determinarea numărului de ordine lexicografică.

Programul care calculează drumul de cost minim este următorul.

Listing HAM.PAS

```
Program HamiltonianDinamic;
const MaxN = 10;
      combinari:array[0..MaxN,0..MaxN] of
                                    Byte=
     ((1,0,0,0,0,0,0,0,0,0,0),
      (1,1,0,0,0,0,0,0,0,0,0),
      (1,2,1,0,0,0,0,0,0,0,0),
      (1,3,3,1,0,0,0,0,0,0,0),
      (1,4,6,4,1,0,0,0,0,0,0),
      (1,5,10,10,5,1,0,0,0,0,0),
      (1,6,15,20,15,6,1,0,0,0,0),
      (1,7,21,35,35,21,7,1,0,0,0),
      (1,8,28,56,70,56,28,8,1,0,0),
      (1,9,36,84,126,126,84,36,9,1,0),
      (1,10,45,120,210,252,210,120,45,
                                  10,1));
type TComb=array[1..MaxN] of Byte;
     TVectorDinamic=array[1..260] of
                                    Byte;
var n,i,j,k:Byte; f:Text;
    d:array[0..MaxN,0..MaxN] of Byte;
    c:array[0..MaxN,1..MaxN] of
                          ^TVectorDinamic;
procedure Citire;
begin
  Assign(f,'HAM.IN'); Reset(f);
  Readln(f,n);
  for i:=0 to n do
    for j:=0 to n do Read(f,d[i,j]);
  Close(f)
end;

function CombinareToNumar(comb:TComb;
                          k:Byte):Longint;
var j,i,nn:Byte; nr:Longint;
begin
  nn:=n; nr:=1;
  for i:=1 to k do
  begin
    for j:=i+1 to k do
      Dec(comb[j],comb[i]);
      for j:=1 to comb[i]-1 do
        Inc(nr,combinari[nn-j,k-i]);
      Dec(nn,comb[i])
  end;
  CombinareToNumar:=nr
end;

procedure NumarToCombinare(nr:Longint;
                k:Byte; var comb:TComb);
var j,i,vl,nn:Byte;
begin
  vl:=0; nn:=n;
  for i:=1 to k  do
  begin
    j:=0;
    while (nr>0) and (nn-j>=k-i) do
    begin
      Inc(j);
      Dec(nr,combinari[nn-j,k-i])
    end;
    Inc(nr,combinari[nn-j,k-i]);
    Dec(nn,j);
    comb[i]:=vl+j;
    Inc(vl,j)
  end
end;

procedure Calcul;
var co,CombNoua:TComb; w,rez:Word;
    elim,t:Byte;
begin                          {initializare}
  for i:=0 to n do
    for j:=1 to n do
      begin
        New(c[i,j]);
        c[i,1]^[j]:=d[i,j]+d[j,0]
      end;
```

```
  for k:=2 to n do
    for i:=0 to n do
      for w:=1 to combinari[n,k] do
{pentru toate combinarile de lungime k}
      begin
        NumarToCombinare(w,k,co);
        c[i,k]^[w]:=255;
        for j:=1 to k do
        begin
          CombNoua:=co;
          elim:=co[j];
          for t:=j to k-1 do
            CombNoua[t]:=CombNoua[t+1];
          rez:=CombinareToNumar
                          (CombNoua,k-1);
          if c[i,k]^[w]>d[i,elim]+
                c[elim,k-1]^[rez] then
            c[i,k]^[w]:=d[i,elim]+
                      c[elim,k-1]^[rez]
        end
      end
end;

Begin
  Citire;
  Calcul;
  Writeln('Drumul minim are lungimea: ',
                              c[0,n]^[1]);
End.
```

Reconstituirea soluției se face fie prin stocarea într-o altă matrice a poziției de proveniență a minimului numerelor $c[i]^[j]$, fie prin reconstituirea (de data aceasta de sus în jos) a soluției optime.

Listing procedura Reconstituie

```
procedure Reconstituie(i,k:Byte;
                              w:Longint);
var rez:Longint; j,elim,t:Byte;
    co,combNoua:TComb;
begin
  if k=1 then Write(w,' 0')
  else
    begin
      NumarToCombinare(w,k,co);
      j:=0;
      repeat
        Inc(j); CombNoua:=co;
        elim:=co[j];
        for t:=j to k-1 do
          mbNoua[t]:=CombNoua[t+1];
        rez:=CombinareToNumar
                          (CombNoua,k-1);
      until c[i,k]^[w]=d[i,elim]+
                      c[elim,k-1]^[rez];
      Write(elim,' ');
      Reconstituie(elim,k-1,rez)
    end
end;
```

Intrare

```
40 1 2 3 41 0 7 10 152 7 0 6 83.10 6 0 204
15 8 20 0
```

Ieșire

```
Drumul minim are lungimea: 29
Drumul este: 0 1 3 2 4 0
```

Intrare

```
30 115 10 1115 0 6 2010 6 0 21 20 2 0
```

Ieșire

```
Drumul minim are lungimea: 37
Drumul este: 0 2 1 3 0
```

Putem reduce cantitatea de memorie folosită, deoarece știm că numerele $c[i, k]$, depind doar de numerele $c[i, k-1]$, adică doar de submulțimile de cardinal k-1 a mulțimii numerelor $j_1, j_2, ..., j_k$.

În locul matricei c vom avea acum doi vectori *c1* și *c2*.

```
c1,c2:array[0..MaxN] of ^TVectorDinamic;
```

Noua procedură `Calcul` este:

Listing procedura Calcul

```
procedure Calcul;var co,CombNoua:TComb;
    w,rez:Word;
    elim,t:Byte;
begin                              {initializare}
  for i:=0 to n do
  begin
    New(c1[i]); New(c2[i]);
    for j:=1 to n do
      c1[i]^[j]:=d[i,j]+d[j,0]
  end;
  for k:=2 to N do
  begin
    for i:=0 to n do
    for w:=1 to combinari[n,k] do
{pentru toate combinarile de lungime k}
    begin
      NumarToCombinare(w,k,co);
      c2[i]^[w]:=255;
      for j:=1 to k do
      begin
        CombNoua:=co;
        elim:=co[j];
        for t:=j to k-1 do
          CombNoua[t]:=CombNoua[t+1];
        rez:=CombinareToNumar
                          (CombNoua,k-1);
```

```
        if c2[i]^[w]>d[i,elim]+
                    c1[elim]^[rez] then
          c2[i]^[w]:=d[i,elim]+
                      c1[elim]^[rez]
      end
    end;
    for i:=0 to n do
    Move(c2[i]^,c1[i]^,Sizeof(c2[i]^))
  end
end;

Begin
  Citire; Calcul;
  Writeln('Drumul minim are lungimea: ',
                              c2[0]^[1])
End.
```

În acest caz, reconstituirea soluției nu se mai face așa de simplu, ea necesitând refacerea matricei *c*.

Se presupune că nu între toate orașele există legături directe. Determinați în acest caz formulele de calcul ale numerelor $c[i, j_1, ..., j_k]$ și implementați într-un program ceea ce ați obținut. Calculați drumul elementar de lungime maximă dintre două noduri ale acestui graf.

Diferență minimă

Se dă un șir de *n* cifre (*n* par). Se cere ca din cifrele date să se formeze două numere (fiecare de lungime $[n/2]$) astfel încât diferența dintre ele să fie minimă.

Restricții

Datele de intrare se citesc dintr-un fișier text având următoarea structură:

- pe prima linie se află numărul *n* (par);
- pe următoarea linie se dau cele *n* cifre, despărțite printr-un spațiu.

Soluție

Presupunem că la un anumit pas al unei soluții optime, punem a *i*-a cifră (din șirul de cifre dat) la primul număr și a *j*-a cifră la cel de al doilea număr și au mai rămas $2k$ cifre $j_1, j_2, ..., j_{2k}$ nefolosite. Construirea celor două numere se face în paralel (deoarece au și lungimi egale), începând cu cifra cea mai semnificativă și continuând spre cifra cea mai nesemnificativă.

Este clar că, deoarece diferența dintre numere este în final minimă, trebuie ca diferența dintre numerele care au cifrele cele mai semnificative a *i*-a, respectiv a *j*-a și care mai folosesc cifrele $j_1, j_2, ..., j_{2k}$, să fie de asemenea minimă, deoarece în caz contrar întreaga diferență nu ar mai fi optimă pentru că s-ar putea face o altă distribuție mai bună a numerelor $j_1, j_2, ..., j_{2k}$.

Să definim deci:

$c[i, j, j_1, j_2, ..., j_{2k}]$ = diferența dintre două numere care încep cu a *i*-a respectiv a *j*-a cifră (dintre cele date) și care mai folosesc și cifrele $j_1, j_2, ..., j_{2k}$.

Soluția problemei se va afla printre numerele $c[i, j, j_1, j_2, ..., j_{N-2}]$.

Fie *a* vectorul în care am citit cifrele de la intrare. Avem:

$$c[i, j, j_1, j_2, ..., j_{2*k}] = \min_{1 \le p,q \le 2k} \{c[j_p, j_q, j_1, j_2, ..., j_{p-1}, j_{p+1}, ..., j_{q-1}, j_{q+1}, ..., j_k] * 10 + a[i] - a[j]\}$$

Această formulă este o aplicație directă a principiului optimalității.

Pentru calcularea acestei formule începem cu:

$c[i, j]=a[i]-a[j]$,

iar apoi calculăm pe $c[i, j, j_1, j_2]$, apoi pe $c[i, j, j_1, j_2, j_3, j_4]$ etc.

Secvența de valori (j_p, j_q) care minimizează expresia de mai sus conduce la soluția optimă.

Compexitatea acestui algoritm este $O(N^5 2^N)$.

Listing IMPART.PAS

```
Program ImpartireDinamic;
const MaxN=10;
      combinari:array[0..MaxN,0..MaxN]
                                of Word =
       { aici inserati constantele de }
       {        la programul anterior }
type TComb=array[1..MaxN] of Byte;
     TVectorDinamic=array[1..260] of
                                Integer;
var n,i,j,k,w:Byte;
    a:array[1..MaxN] of Byte;
    c:array[1..MaxN,1..MaxN,
     0..MaxN div 2] of ^TVectorDinamic;
    f:Text;
    min:Integer;
    nr1,nr2,co:TCombinare;
    ii,jj,ww:Byte;

procedure Citire;
begin
 Assign(f,'NUMERE.IN');
 Reset(f);
 Readln(f,n);
 for i:=1 to n do
  Read(f,a[i]);
 Close(f)
end;

{     aici vor fi inserate procedurile }
{ CombinareToNumar si NumarToCombinare }
{              din programul precedent }

function Exista(n1,n2:Byte;comb:TComb;
                    lung:Byte):Boolean;
{returneaza true daca unul din numerele}
{       n1 si n2 se gaseste in vectorul}
{                 comb de lungime lung}
```

```
var c:Byte;
begin
 for c:=1 to lung do
  if comb[c] in [n1,n2]
   then begin
          Exista:=true; Exit
        end;
 Exista:=false
end;

procedure Calcul;
var co,CombNoua:TComb;
    w,rez:Word;
    p,q,t,min,max:Byte;
begin                          {initializare}
 for i:=1 to n do
  for j:=1 to n do
   for k:=0 to (n-2) div 2 do
    begin
     New(c[i,j,k]);
     c[i,j,0]^[1]:=a[i]-a[j]
    end;
 for k:=1 to (n-2) div 2 do
  for i:=1 to n do
   for j:=1 to n do
   if i<>j then
    for w:=1 to combinari[n,2*k] do
    begin
     NumarToCombinare(w,2*k,co);
     c[i,j,k]^[w]:=Maxint;
     if not Exista(i,j,co,2*k) then
     for p:=1 to 2*k do
      for q:=1 to 2*k do
      if (p<>q) then
      begin
       CombNoua:=co;
       if p<q then
        begin min:=p; max:=q end
       else
        begin min:=q; max:=p end;
        for t:=min to max-2 do
         CombNoua[t]:=CombNoua[t+1];
         for t:=max-1 to 2*k-2 do
          CombNoua[t]:=CombNoua[t+2];
         rez:=CombinareToNumar
                    (CombNoua,2*k-2);
         if Abs(c[i,j,k]^[w])>
              Abs(c[co[p],co[q],k-1]^
                  [rez]*10+a[i]-a[j])
         then c[i,j,k]^[w]:=
                  c[co[p],co[q],k-1]^
                  [rez]*10+a[i]-a[j];
    end
   end
end;
```

```
procedure Reconstituie(i,j,k:Byte;
                       w:Word);
var co,combNoua:TComb;
    min,max,t,p,q:Byte;
    rez:Word;
begin
 nr1[k+1]:=a[i];
 nr2[k+1]:=a[j];
 if k>0 then
 begin
  NumarToCombinare(w,2*k,co);
  if not Exista(i,j,co,2*k) then
  begin
   p:=0;
   repeat
    Inc(p); q:=0;
    repeat
     Inc(q);
     if p<>q then
     begin
      CombNoua:=co;
      if p<q then
       begin
         min:=p;
         max:=q
       end
      else
       begin
        min:=q;
        max:=p
       end;
      for t:=min to max-2 do
       CombNoua[t]:=CombNoua[t+1];
      for t:=max-1 to 2*k-2 do
       CombNoua[t]:=CombNoua[t+2];
      rez:=CombinareToNumar
                    (CombNoua,2*k-2)
     end;
    until (q=2*k) or (c[i,j,k]^[w]=
                  c[co[p],co[q],k-1]^
                  [rez]*10+a[i]-a[j]);
   until (c[i,j,k]^[w]=c[co[p],co[q],
                 k-1]^[rez]*10+a[i]-a[j]);
   Reconstituie(co[p],co[q],k-1,rez)
  end
 end
end;

Begin
 Citire;
 Calcul;
 min:=Maxint;
 for w:=1 to combinari[n,2] do
 begin
  NumarToCombinare(w,n-2 ,co);
```

```
  for i:=1 to n do
   for j:=1 to n do
    if i<>j then
     if not exista(i,j,co,n-2) then
      if Abs(min)>Abs(c[i,j,(n-2) div
                              2]^[w])then
      begin
       min:=c[i,j,(n-2) div 2]^[w];
       ii:=i; jj:=j; ww:=w
      end
 end;
 Writeln('Diferenta minima: ',
                                  min);
 Reconstituie(ii,jj,(n-2) div 2,ww);
 Write('Primul numar:');
 for i:=1 to n div 2 do
  Write(nr1[i]);
 Writeln;
 Write('Al doilea numar:');
 for i:=1 to n div 2 do
  Write(nr2[i])
End.
```

Intrare	Ieșire
4	Diferenta minima: 1
1 1 9 2	Primul numar: 19
	Al doilea numar: 21

Intrare	Ieșire
6	Diferenta minima: 2
9 9 2 0 1 1	Primul numar: 201
	Al doilea numar: 199

Ca și în cazul problemei drumului hamiltonian se poate implementa un program care să nu mai necesite parametrul *k* în matricea *c*. În acest caz reconstituirea numerelor care au generat această soluție s-ar face mai greu.

Concluzie

Ceea ce am propus în aceste pagini este o modalitate specială de a rezolva problemele NP-complete (sau chiar unele probleme polinomiale despre care nu suntem siguri că algoritmul propus este optim) folosind metoda programării dinamice. Ideea algoritmului este identică în ambele probleme rezolvate mai sus, de aceea, cu ajutorul ei se pot soluționa o serie de probleme având ordin de complexitate exponențial.

Bibliografie

1. Colecția *Journal of the ACM*
2. Mihai Oltean, *Proiectarea și implementarea algoritmilor*, Editura Computer Libris Agora, Cluj-Napoca, 1999.

Mihai Oltean este doctorand al Universității "Babeș-Bolyai" din Cluj și poate fi contactat pe adresa moltean@cs.ubbcluj.ro.

News...

Microsoft Windows 2000

Mult așteptatul eveniment al lansării oficiale a noului sistem de operare de la Microsoft avut loc pe data de 17 februarie, în jurul orei 19.30 (ora Romaniei), când Bill Gates a prezentat noul sistem de operare, într-un show transmis în direct și pe Internet. Bill Gates a numit sistemul de operare Windows 2000 ca primul în cea mai importantă generație de produse pe care le-a lansat vreodată Microsoft. Gates a mai spus că utilizatorii sistemului de operare Windows 2000 vor beneficia de o stabilitate mai mare, de scalabilitate, performanță și avantaje în ceea ce privește prețul față de platformele alternative. Într-un interviu acordat televiziunii CNBC, Gates a admis că unii utilizatori vor avea probleme de compatibilitate, în special cei care vor face upgrade de la Windows NT, dar a mai spus că aproape toate firmele vor adopta noul sistem de operare. "Cred că este cinstit să spunem că unul din patru clienți vor avea unele probleme când vor face upgrade, dar beneficiile sunt foarte mari", a spus Gates. La lansare au fost prezenți și actorul Patrick Stewart, cunoscut din filmul Star Trek: Next Generation, și cântărețul rock Carlos Santa împreună cu formația sa. Gates a mai declarat că dezvoltarea sistemului de operare Windows 2000 a costat mai mult de 1 miliard de dolari și a implicat colaborarea cu mai mult de 1.000 de parteneri.

Microsoft Windows Me

Compania Microsoft a hotărât să denumească viitorul sistem de operare pentru utilizatorii casnici Windows Me, aceasta fiind o prescurtare de la Microsoft Windows Millenium Edition. Cunoscut până acum sub numele de cod Millenium, Windows Me, la fel ca și Windows '98, va fi preinstalat pe calculatoarele noi bazate pe Windows. După ce Bill Gates a anunțat că Windows '98 va fi ultimul sistem de operare al companiei bazat pe DOS, Microsoft s-a contrazis și a lansat anul trecut Windows '98 Second Edition, anunțându-și totodată și planurile pentru Millenium.

BACALAUREAT 2000

Primul examen de Bacalaureat care prevede pentru absolvenții claselor de informatică și o probă scrisă la disciplina de specialitate pune multe întrebări în fața elevilor și profesorilor. Ce fel de subiecte se pot formula pentru acest examen? Cum poate arăta o lucrare scrisă la informatică? Ginfo este iar alături de voi! Vă prezentăm un posibil subiect de Bac și rezolvarea corespunzătoare.

Subiectul 1

1. Scrieți pseudocodul algormitmului care simulează împărțirea a două numere naturale astfel încât câtul și restul să fie numere întregi.

Soluție

Dacă există posibilitatea ca numerele să fie egale cu `0`, pseudocodul trebuie să conțină două structuri alternative care să afișeze câtul `0` și restul `0` dacă `m=0`, respectiv mesajul `Nu putem imparti cu 0`, dacă `n=0`.

Pentru numere strict pozitive, împărțirea celor două numere se poate simula repetând operația de scădere în felul următor:

```
citește m,n (m,n>0)
dacă m = 0
  atunci scrie 0, 0
  altfel
    dacă n = 0
      atunci scrie Nu putem imparti cu 0
      altfel
        d ← 0
        cât timp m ≠ n execută
          m ← m - n
          d ← d + 1
        sfârșit cât timp
        scrie d, m
    sfârșit dacă
sfârșit dacă
```

2. Secvența următoare reprezintă pseudocodul problemei determinării minimului dintr-un șir de numere întregi. Găsiți greșeala și corectați algoritmul!

```
citește n
pentru i=1,n execută
citește a[i]
pentru i=1,n execută
  dacă a[i] < min
    atunci min ← a[i]
  sfârșit dacă
sfârșit pentru
```

Soluție

Greșeala care trebuie observată este lipsa instrucțiunii de inițializare a variabilei `min`. Corectarea este simplă: fie se inițializează `min` cu `a[1]`, fie cu o valoare suficient de mare, față de care în mod sigur vom avea elemente având valori mai mici în șir, astfel oferind posibilitatea înlocuirii sale cu o valoare din șir.

3. Prezentați comparativ structura repetitivă cu test inițial și cea cu test final. Transcrieți structura repetitivă cu test final în forma unei structuri repetitive cu test inițial.

Soluție

- Ambele structuri sunt repetitive, cu număr necunoscut de pași.
- În ambele structuri, cel puțin una din variabilele ce apar în expresia logică (în condiție) trebuie să se modifice în corpul ciclului, altfel se generează un ciclu infinit.
- Dacă valoarea de adevăr a condiției de ciclare din structura repetitivă cu test inițial este falsă la prima evaluare a ei, corpul acestuia nu se va executa nici măcar o dată, în timp ce corpul structurii repetitive cu test final, se execută cel puțin o dată, indiferent de valoarea de adevăr a condiției de ciclare.

Structură repetitivă cu test inițial

```
cât timp condiție execută
   corp
sfârșit cât timp
```

Structură repetitivă cu test final

```
execută
   corp
până când condiție
```

Transcrierea structurii repetitive cu test final în structură repetitivă cu test inițial

```
corp
cât timp not condiție execută
   corp
sfârșit cât timp
```

4. Explicați rolul stivei în contextul executării subprogramelor.

Soluție

Dacă într-un program există subprograme, în momentul apelului acestora mediul efectuează următoarele operații, alocând spațiu pe *stiva de execuție* pentru diferite informații din care se compune contextul apelului:

- se salvează adresa instrucțiunii care se va executa după terminarea execuției subprogramului; astfel va fi posibilă revenirea la instrucțiunea de după "locul apelului";
- se alocă spațiu pentru parametrii formali;
- se copiază valorile parametrilor actuali transmiși prin valoare în spațiul alocat parametrilor formali corespunzători;
- se copiază adresele parametrilor actuali transmiși prin referință în spațiul alocat parametrilor formali corespunzători;
- se alocă spațiu pentru variabilele locale.

Subiectul 2

1. Scrieți un subprogram care returnează o valoare logică, reprezentând valoarea de adevăr a proprietății: *tabloul bidimensional pătratic, transmis ca parametru are pe cele două diagonale (principală și secundară) elemente distincte.*

Soluție

Știind că elementele de pe diagonala principală au expresiile de indice egali, de exemplu `tab[i,i]` iar cele pe diagonala secundară au forma `tab[i,dim-i+1]` (dacă indiicii pornesc de la `1` - limbajul *Pascal*), respectiv `tab[i,dim-i-1]` (dacă indiicii pornesc de la `0` - limbajul *C*) funcția se implementează în felul următor.

Varianta Pascal:

```
Function distincte(tab:matrice;
                   dim:Byte):Boolean;
var i,j:Byte; dist:Boolean;
begin
  dist:=true; i:=1;
  while dist and (i<dim) do
  begin
    j:=i+1;
    while dist and (j<=dim) do
    begin      { distincte pe diag. princ.? }
      dist:=dist and (tab[i,i]<>tab[j,j]);
                 { distincte pe diag. sec.? }
      dist:=dist and
            (tab[i,dim-i+1]<>tab[j,dim-j+1]);
      { distincte pe diag. princ. si sec.? }
      dist:=dist and
                 (tab[i,i]<>tab[j,dim-j+1]);
      Inc(j)
    end;
    Inc(i)
  end;
  distincte:=dist
end;
```

Varianta C:

```
int distincte(int tab[][],int dim){
  int dist=1;
  for (int i=0;i<dim-1 && dist;i++){
    for (int j=i;j<dim && dist;j++){
      // distincte pe diag. princ.?
      dist&=(tab[i,i]!=tab[j,j]);
      // distincte pe diag. sec.?
      dist&=(tab[i,dim-i-1]!=tab[j,dim-j-1]);
       // distincte pe diag. princ. si sec.?
      dist&=(tab[i,i]!=tab[j,dim-j-1]);
    }
  }
  return dist;
}
```

Deoarece nu am dorit să scriem o funcție recursivă, am folosit o variabilă auxiliară `dist`, pentru a putea folosi valoarea curentă a acesteia (având semnificația "distincte"). Datorită modului în care această variabilă conține în valoarea ei informația despre proprietatea căutată, putem părăsi ciclurile imediat ce s-au găsit două elemente egale.

2. Scrieți un subprogram recursiv pentru calcularea sumei primelor `n` numere naturale.

Soluție

Din moment ce subprogramul trebuie să returneze o singură valoare (suma), vom scrie o funcție. Formula pe care o vom implementa este următoarea:

`Suma(1)=1, Suma(n)=n+Suma(n-1)`

Varianta Pascal:

```
Function suma(n:Byte):Longint;
begin
  if n=1 then suma:=1
         else suma:=n+suma(n-1)
end;
```

Varianta C:

```
long suma(int n){
  if (n=1) return 1;
  else return n+suma(n-1);
}
```

Subiectul 3

La o benzinărie sosesc `n` mașini. Proprietarul dorește să minimizeze timpul total de așteptare a celor care doresc să cumpere benzină. Scrieți un program comentat, care stabilește ordinea în care se vor servi mașinile știind că timpul de servire a fiecărei mașini este direct proporțional cu cantitatea de benzină cumpărată. Afișați ordinea servirii și timpul total de așteptare.

Exemplu

Intrare	*Ieșire*
n=5	Ordinea:
m1=2	m2
m2=1	m1
m3=4	m4
m4=3	m3
m5=5	m5
	Timpul total minim de asteptare: 20

Soluție

Problema se va rezolva cu metoda *greedy*, deoarece timpul total de așteptare se minimizează dacă mașinile se vor alimenta în ordinea crescătoare a cantității de benzină cumpărată. Pentru ordonare s-a folosit o metodă conform căreia se stabilește maximul șirului și se așează pe ultima poziție, după care dimensiunea șirului se micșorează (temporar), pentru a putea alege următorul cel mai mare element, așezându-l și pe acesta pe poziția sa în șirul ordonat, și așa mai departe.

Pentru a putea afișa ordinea servirii mașinilor folosind numărul lor de ordine în șirul inițial (neordonat), păstrăm numărul de ordine inițial într-un câmp `nr` a articolelor de tip `masina`.

Implementarea Pascal este următoarea:

Listing BENZINARIE.PAS

```
Program benzinarie;
const MaxNr=50; {numar max. de masini}
type masina=record
             cant_benz:Integer;
             {cantitatea de benzina}
             nr:Byte
           end;
var n,i,j:Byte; {nr = numar de masini}
    m:array[1..MaxNr] of masina;
    timp:array[1..MaxNr] of Integer;
    {timpul de asteptare al masinilor}
    timp_total:Longint;
                {timpul total de asteptare}

procedure ordonare(dim:Byte);
var max,aux:masina;
    i,poz:Byte;
begin
  while dim>0 do
  begin
    max:=m[1];
    poz:=1;          { pozitia maximului }
    for i:=1 to dim do
      if m[i].cant_benz>max.cant_benz
      then begin
             max:=m[i]; poz:=i
           end;
      aux:=m[poz];
      m[poz]:=m[dim];
      m[dim]:=aux;
      Dec(dim)
          {vom cauta intr-un sir mai scurt}
  end
end;

Begin
  Write('n='); Readln(n);
  Write('Cantitatile de benzina care');
  Writeln(' se cumpara:');
  for i:=1 to n do
  begin
    Read(m[i].cant_benz); m[i].nr:=i
  end;
  ordonare(n);
```

```
  Writeln('Ordinea:');
  for i:=1 to n do
    Writeln('m',m[i].nr);
  timp[1]:=0;
  timp_total:=0;
  for i:=2 to n do
  begin
    timp[i]:=timp[i-1]+m[i-1].cant_benz;
    Inc(timp_total,timp[i])
  end;
  Write('Timpul total minim de ');
  Writeln('asteptare: ',timp[n])
End.
```

Versiunea C este prezentată în continuare:

Listing BENZINARIE.CPP

```
include <stdio.h>
#define MaxNr 50 //numarul maxim de masini

struct masina{
  int cant_benz; //cantitatea de benzina
  int nr;
} m[MaxNr];
int n,timp[MaxNr];
int timp_total; //timpul total de asteptare

void ordonare(int dim){
  masina max,aux;
  int poz; //pozitia maximului
  while (dim){
    max=m[poz=0];
    for (int i=1;i<dim;i++)
      if (m[i].cant_benz>max.cant_benz)
        max=m[i], poz=i;
    aux=m[poz];
    m[poz]=m[dim-1];
    m[dim---1]=aux;
  }
}

void main(void){
  printf("n=");
  scanf("%d",&n);
  printf("Cantitatile de benzina ");
  printf("care se cumpara:\n");
  for (int i=0;i<n;i++){
    scanf("%d",&m[i].cant_benz);
    m[i].nr=i;
  }
  ordonare(n);
  puts("Ordinea:");
  for (i=0;i<n;i++)
    printf("m%d\n",m[i].nr+1);
  timp[0]=timp_total=0;
  for (i=1;i<n;i++)
    timp[i]=timp[i-1]+m[i-1].cant_benz,
    timp_total+=timp[i];
  printf("Timpul total minim de");
  printf("asteptare: %d\n", timp_total);
}
```

Subiectul 4

Gigel vine acasă de la școală și își despachetează rucsacul. Își așează cărțile pe birou într-o stivă. Scrieți un subprogram comentat care simulează operația de creare a stivei de cărți. Subprogramul trebuie să returneze pointerul q care indică stiva creată. Pentru fiecare carte se va citi titlul. Numărul cărților nu se cunoaște.

Soluție

Deoarece numărul cărților nu se cunoaște, vom răspunde în program la întrebarea: `Mai sunt carti in rucsac?` (variabila `raspuns`). Condiția din instrucțiunea repetitivă **while** verifică dacă răspunsul la întrebare este `'D'` (DA), sau `'N'` (NU) pentru a avea posibilitatea de a continua, după caz cu inserarea unui element nou în stiva de cărți, sau pentru a încheia crearea. Pentru a asigura acceptarea răspunsului atât cu litere mari cât și mici, acesta se transformă în literă mare cu ajutorul funcției `Upcase` (pentru *Pascal*), respectiv `toupper()` (pentru C). Deoarece crearea stivei se realizează în subprogram, inițializarea pointerului care va indica capul ei după creare, poate fi realizat fie în subprogram, fie în programul principal.

Varianta Pascal:

Presupunem că stiva are următorul tip:

```
type carti=^carte;
     carte=record
             titlu:string[10];
             urm:carti
           end;
```

Parametrul formal q se transmite prin referință, pentru a asigura accesul la stivă în afara subprogramului. Subprogramul de creare poate fi scris în felul următor:

```
Procedure creare(var q:carti);
var raspuns:Char;
    p:carti;
begin
  q:=nil;
  Write('Exista cel putin o carte? ');
  Readln(raspuns);
  while Upcase(raspuns)='D' do
  begin
    New(p);
    Write('Titlul: ');
```

```
    Readln(p^.titlu);
    p^.urm:=q;
    q:=p;
    Write('Mai exista carti in rucsac? ');
    Readln(raspuns)
  end
end;
```

Varianta C:

Presupunem că stiva are următorul tip:

```
struct carti{
  char titlu[10];
  urm *carti;
}
```

Vom scrie o funcție care va returna o referință spre capul stivei. Suprogramul de creare poate fi scris în felul următor:

```
carti* creare(void){
  char raspuns;
  carti *p, *q=NULL;
  printf("Exista cel putin o carte? ");
  fscanf("%c\n",&raspuns);
  while (toupper(raspuns)=='D'){
    p=new carti;
    printf("Titlul:"); scanf("%s",p->titlu);
    p->urm=q; q=p;
    printf("Mai exista carti in rucsac? ");
    fscanf("%c\n",&raspuns);
  }
  return q;
}
```

Barem

Subiectul 1: 30 puncte
- 1) 7,5 puncte
- 2) 7,5 puncte
- 3) 7,5 puncte
- 4) 7,5 puncte

Subiectul 2: 20 puncte
- 1) 10 puncte
- 2) 10 puncte

Subiectul 3: 20 puncte

Subiectul 4: 20 puncte

Se acordă 10 puncte din oficiu.

TOTAL: 100 puncte

News...

AMD Athlon 850

Mai multă performanță și frecvență de ceas mai ridicată - dedicată acestor țeluri, compania Advanced Micro Devices și-a anunțat lansarea noilor procesoare Athlon la 850 MHz pentru desktop-uri. Odată cu această lansare, AMD revine pe prima poziție în ceea ce privește viteza de procesare a cipurilor aflate pe piață în acest moment. Cel mai rapid procesor Pentium III (produs de eterna rivală Intel) funcționează la o viteză de ceas de numai 800 MHz, adică cu 50 MHz mai puțin. Noul Athlon are un preț de vânzare de 849 de dolari la cantități de cel puțin 1000 de unități și va fi disponibil în mai multe tipuri de calculatoare. Printre producătorii care au început să îl folosească se numără Compaq Computer, IBM și Gateway. Cipul Athlon este produs în unitatea AMD din Austin, Texas, cu ajutorul tehnologiei de 0,18 microni.

Reacția Intel nu se lasă așteptată; viitorul procesor Pentium III va funcționa la o frecvență de 866 MHz.

Steve Ballmer

Steve Ballmer este noul Președinte și Director Executiv al companiei Microsoft, preluând astfel conducerea efectivă a gigantului din industria software. Fondatorul companiei și acționarul principal, Bill Gates, a renunțat la aceste funcții, el rămânând Președinte al Consiliului de Administrație și Proiectant Software Principal.

Bill Gates își va dedica timpul lansării viitoarei generații de platforme și servicii Windows pentru Internet.

Aceste schimbări au fost anunțate având în vedere schimbarea de strategie care se va impune în urma lansării platformei Windows 2000, care, așa cum afirmă compania, va aduce schimbări majore în industria informatică.

Bill Gates și Steve Ballmer au prezentat strategia care va transforma compania Microsoft, în jurul noilor concepte Internet User Experience (IUE) și Next Generation Windows Services (NGWS).

Așteptăm scrisorile voastre!

POȘTA redacției

La cererea cititorilor am hotărât ca, începând cu acest număr, să redeschidem rubrica Poșta redacției. Vă reamintim că această rubrică nu poate supraviețui fără scrisorile voastre, deci ne exprimăm speranța că vom primi sute de mesaje pe adresa office@ginfo.ro. Totodată, sperăm că această rubrică va contribui la o mai bună comunicare între cititori și redacție, pentru ca GInfo să devină o revistă pe placul tuturor. Vă asigurăm că toate ideile, opiniile și criticile voastre vor fi luate în considerare chiar dacă spațiul alocat acestei rubrici nu va fi suficient pentru a răspunde la toate scrisorile. Și acum să vedem ce am găsit în cutia poștală...

lect. dr. Marin Vlada

Apreciez faptul că în GInfo nr. 1/2000 a apărut articolul *Grafica TURTLE și aplicații în Pascal* de Ion Gelu Tomșa.

Subiectul articolului este interesant pentru elevi, studenți și profesori. După ce am parcurs conținutul articolului, deși în interior se face referire la cartea *The algorithmic Beauty of Plants* de P. Prusinkiewicz, A. Lindenmayer, am rămas surprins când la sfârșitul articolului nu am găsit bibliografie și anume este omisă cartea *Grafica pe calculator în limbajele Pascal și C , vol.II, Aplicații*, Ed. Tehnică, 1992 de M. Vlada, A. Posea, I. Nistor, C. Constantinescu, unde în capitolul 6 *Aplicații în geometria TURTLE* (pag. 93-133) se tratează pe larg acest subiect.

Fac această observație după ce am recunoscut propoziții, fraze și idei din acest capitol redactat de subsemnatul care este titular al cursului de *Grafică pe calculator* la *Colegiul de Informatică*.

După părerea mea, referirea și trimiterea la cartea menționată este necesară ținând seama că simpla prezentare a unor surse de proceduri nu poate suplini descrierea și înțelegerea conceptelor complexe implementate.

Felicit redacția pentru subiectul prezentat cu rezerva celor prezentate mai sus.

Din aceste motive, rog redacția să găsească forma prin care să se precizeze observația principală a subsemnatului.

Ne cerem scuze pentru omisiune, mai ales că autorul, Ion Gelu Tomșa, a trimis articolul la redacție cu bibliografia completă. Sperăm că publicând scrisoarea dumneavoastra am reparat, într-o oarecare măsură, greșeala.

Mihai Pătrașcu

De ce nu se acceptă la Bursele Agora și surse FORTRAN, dacă tot se acceptă aberații ca *Java* sau *Pascal*?

Probabil pentru că majoritatea programatorilor consideră FORTRAN-ul ca fiind o aberație mult mai mare decât limbajele Java și Pascal. Lăsăm totuși problema deschisă sperând că cititorii noștri își vor exprima punctul lor personal de vedere.

Angel Popescu

Vă interesează să publicați sau să distribuiți (doritorilor) prin intermediul site-ului dvs. un articol detaliat despre programarea aplicațiilor *Windows* multimedia în *Visual C++* și *Visual Basic* cu *Microsoft DirectX 7.0*?

Apelez la această metodă, deoarece *DirectX* este un produs nelipsit din "baza de date" a oricărui programator *Windows*, însă în România această tehnică nu este foarte dezvoltată, folosindu-se încă de către elevi și studenți programarea pe 16 biți care nu mai este de actualitate, rezultând astfel un soft românesc cu pretenții scăzute.

Nu putem să ne pronunțăm asupra publicării unui articol înainte să îl citim. Totuși, ideea este, fără îndoială, interesantă. Sperăm să găsim o modalitate de colaborare în viitor.

Adrian Vicovan

Bursele Agora mi s-a părut un concurs interesant și, ca urmare, am participat și eu. Sper că și pe viitor veți organiza astfel de concursuri pentru tineri.

Sperăm și noi că Bursele Agora va deveni un concurs tradițional organizat de Gazeta de Informatică.

S. M. E.

COMPUTER PRESS AGORA

PC REPORT

ECHIPAMENTE · SISTEME · APLICATII · RETELE · INTERNET

Vol. 9, Nr. 1. Ianuarie 2000 LEI 20.000

Din acest număr:

- **Unelte de modelare UML**
- **Introducere în Python**
- Ghid anti-hacker
- Solid Edge 7
- Standarde GNU
- XML prin SAX
- PC Concrete

www.pcreport.ro

Revista
specialiștilor
în tehnologia informației și comunicațiilor

tehnologia informației și comunicat

Cure-all?

For a great night's sleep, try Compaq Deskpro. It relieves the headaches of day-to-day PC management with powerful features like the new Compaq Insight Manager LC. It's the industry's first integrated tool for managing workgroups of PCs from a simple, browser-based interface. So you spend less time playing doctor and more time getting real work done. Just another healthy dose of exceptional value from Compaq - and another reason Deskpro is number one. For more therapeutic thoughts, visit www.compaq.com/deskpro, or contact us at phone: +40-1-3033100

intel inside
pentium III

COMPAQ
prin distribuitorul autorizat

OMNILOGIC BGS
135, Bucureşti-Ploieşti Road,
Bucharest 1, Romania
PO BOX 18-131
Tel: +40-1-303 31 00
Fax: +40-1-303 31 64
www.omnilogic.ro

Intel, the Intel Inside logo and Pentium are registered trademarks of Intel Corporation

Martie 2000
Vol. 10/3

Lei 15.000

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

ROLUL DECIZIEI
în programarea dinamică

CONCURSURI

Rezultate Runda 4

Soluții Runda 4
Propuse Runda 6

SERIAL

Pagini Web cu XML 1.0
Introducere în teoria grafurilor

BABEL

VIRUȘI

Internet
Retele de calculatoare si mediul Internet

Bac: rezolvarea variantei nr. 18 din cartea Bac Info' 2000

ACTUAL

Procesoare la 1 Ghz

COMPUTER PRESS AGORA

PC MAGAZINE ROMÂNIA

Din acest număr:

- **Windows 2000**
- **Chat-ul productiv**
- **Plăci grafice**
- **Sănătatea și Web-ul**

www.pcmagazine.ro

Revista utilizatorilor de calculatoare personale

COMPUTER PRESS AGORA

Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

Vol. 10, Nr. 3/2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Cristian Cadar (București)
Sorin Groza (Aiud)
Ilie Vieru (Bacău)
Adrian Monea (Cluj)
Flaviu Cibu (Sibiu)
Adrian Bugariu (Sibiu)
Tehnoredactare:
Mihai Uricaru, Călin Borz
Corectura:
Carmen Bucur
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8, Cluj-Napoca, 3400
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Gabriela Bucșa (gbucsa@agora.ro)
Gabriel Piscupescu (gpiscup@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Anamaria Pascu (apascu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Mihaela Szabo (mszabo@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

concurs.. PAG. 5

Rezultatele rundei a patra

Clasament

Problemele ultimei runde

internet.. PAG. 29

REȚELE de calculatoare și mediul INTERNET

Marin VLADA

bac.. PAG. 12

BACALAUREAT 2000

Al doilea bac în informatică

actual.. PAG. 37

Procesoare la 1 GHZ

Adrian MONEA
Mihai SCORȚARU

focus.. PAG. 17

Rolul DECIZIEI în programara DINAMICĂ

Ilie VIERU

babel.. PAG. 40

VIRUȘI

Flaviu CIBU
Adrian BUGARIU

serial.. PAG. 20

Introducere în teoria GRAFURILOR

Cristian CADAR

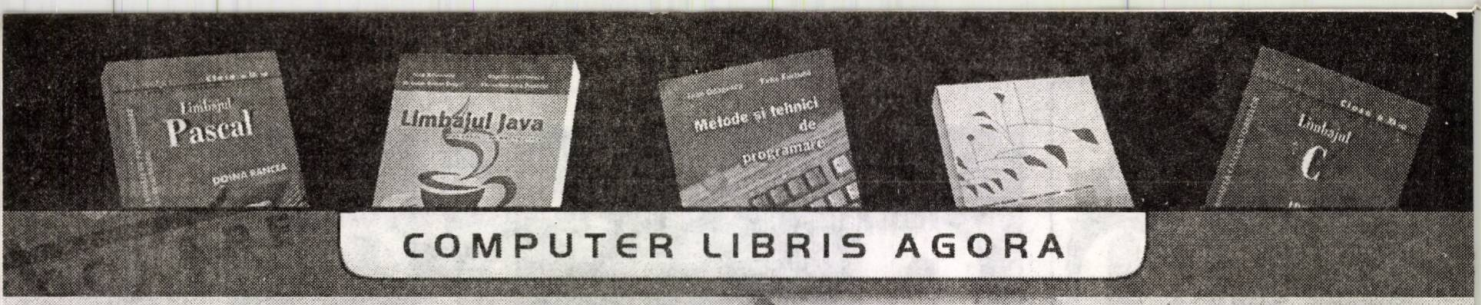
Pagini WEB cu XML 1.0

Mihai SCORȚARU

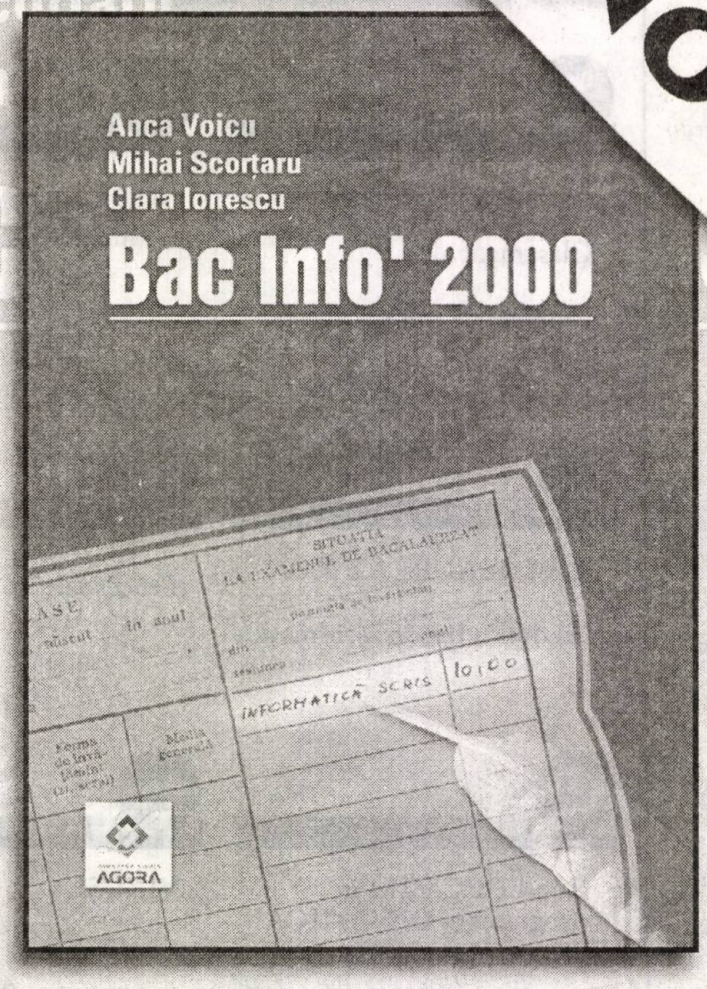
begin.. PAG. 45

Ghiocelul, mărțișorul și ALGORITMII

Sorin GROZA



NOUTATE



În această cărticică veți găsi subiecte de Bac la informatică și nu doar atât.
În partea a doua, vă punem la dispoziție posibile răspunsuri, soluții ale problemelor. Deoarece limbajul de programare este la alegere, depinde de candidat pe care îl preferă. Acolo unde răspunsul depinde de limbajul ales, veți găsi soluțiile corespunzătoare prezentate și în Pascal și în C.
La sfârșitul cărții sunt patru "variante" de subiecte propuse ale căror rezolvări le veți putea citi și studia în Gazeta de Informatică.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

AGORA

Cartea de ALGORITMI

Cum se explică neliniștea noastră, acea neliniște care ne cuprinde în clipa în care vedem o carte apărută în străinătate, o carte pe care o dorim, știm că ne-ar fi extrem de utilă - dar care nu ne este accesibilă? E firesc... veți spune. În momentul respectiv, o visăm pe raftul de cărți, vrem s-o studiem imediat... Și noi, profesorii, ne gândim și la elevii, studenții noștri, care la rândul lor, de asemenea, ar găsi multe informații utile în acea carte și - citind-o - ar putea să-și completeze cunoștințele.

Una din aceste cărți este cea a cărei copertă ați văzut-o în mai multe numere ale revistei noastre, deoarece editura *Computer Libris Agora* a decis s-o publice în limba română. Editura a pornit cu entuziasm proiectul traducerii și editării cărții *"Introduction to Algoritms"*, scrisă de *Thomas H. Cormen, Charles E. Leiserson* și *Ronald L. Rivest*, și editată la *MIT Press, Massachussetts* în Statele Unite.

În anii care au trecut, această carte a fost tradusă în mai multe limbi. Cum putea tocmai România să nu le ofere tinerilor informaticieni tradusă în limba română? Cititorii cărților de acest fel, evident - vorbesc engleza bine, dar în Statele Unite această carte costă 67 de dolari... Nici un profesor nu poate să indice o asemenea bibliografie!

Primul contact cu MIT Press l-am stabilit în toamna anului 1998. Mulțumim doamnei *Cristina Sanmartin* (MIT Press) că ne-a încurajat și a intermediat semnarea contractului pentru traducere și editare în limba română. Acum suntem deja în 2000. A durat traducerea, tehnoredactarea și tipărirea. Pentru apariția în limba română a cărții, adresez mulțumiri profesorilor, studenților și prietenilor care au sacrificat multe ore din timpul lor liber, pentru ca dumneavoastră să puteți ține această carte în mână.

Domnul *conf. dr. Horia F. Pop*, de la Universitatea "Babeș-Bolyai", Facultatea de Matematică și Informatică, a coordonat traducerea, a stabilit modul de traducere a termenilor de specialitate noi, începând cu index-ul, pentru a asigura o traducere coerentă, omogenă și pe cât posibil unitară. De asemenea, a pregătit și coordonat utilizarea sistemului LaTeX.

Am lucrat cu cadre didactice universitare de specialitate deoarece o asemenea carte nu se traduce doar cu lingviști, este nevoie de specialiști din domeniu care înțeleg foarte bine conținutul științific și sunt de acord să reformuleze anumite părți pentru a exprima ceea ce de fapt autorul a dorit să comunice. Mulțumim domnilor (în ordine alfabetică) lector dr. Paul Blaga, profesor dr. Florian Mircea Boian, drd. Liana Bozga, profesor dr. Zoltan Kasa, lector drd. Ioan Lazar, lector drd. Simona Motogna, asistent drd. Virginia Niculescu, profesor dr. Bazil Pârv, lector dr. Radu Trâmbițaș, de la Universitatea "Babeș-Bolyai" din Cluj, domnului profesor Liviu Negrescu de la Universitatea Tehnică din Cluj, domnului profesor dr. Horia Georgescu de la Academia de Studii Economice, doamnei profesor dr. Luminița State de la Universitatea din Pitești, doamnei lector drd. Cristina Vertan, de la Universitatea București și nu în ultimul rând studenților Adrian Monea și Mihai Scorțaru, de la Universitatea Tehnică din Cluj.

Cum aceasta este prima carte editată de Computer Libris Agora în LaTeX, tehnoredactarea a constituit și ea un efort. În special pregătirea desenelor a ridicat multe probleme, graficianul nostru Dan Crețu lucrând nenumărate ore la realizarea acestora. Într-o carte de asemenea dimensiuni și complexitate cu siguranță se strecoară și unele greșeli. Am dori ca acestea să fie eliminate din viitoarele ediții ale lucrării. Din acest motiv, vă rugăm să ne transmiteți observațiile și sugestiile Dumneavoastră pe adresa de e-mail:

clara@cs.ubbcluj.ro

sau pe adresa editurii:

Editura Computer Libris Agora
Str. Universității nr. 8
3400 Cluj Napoca
Tel.: 064-192422

Urmând exemplul american, erata va fi disponibilă pe Internet. Cei care doresc să intre în posesia ultimei erate pot trimite un mesaj la adresa

algoritmi@agora.ro

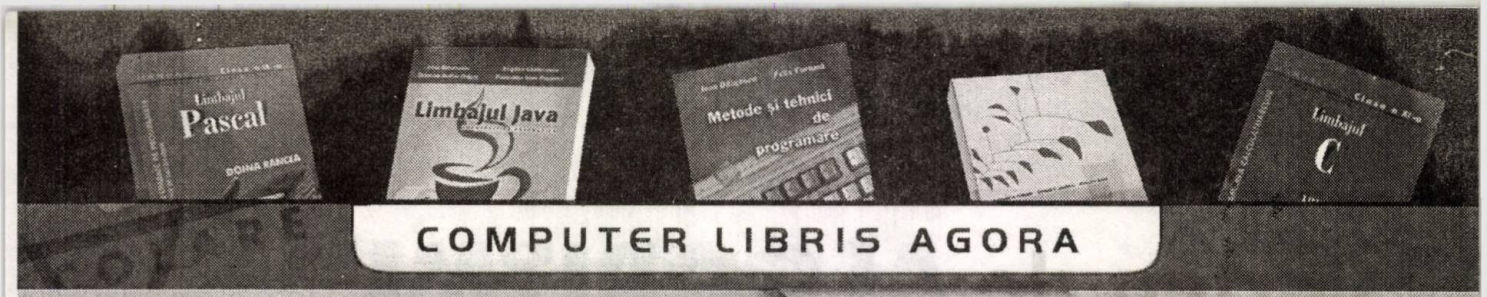
conținând expresia

```
erata algoritmi
```

în câmpul `Subject` și vor primi automat ultima versiune în format electronic.

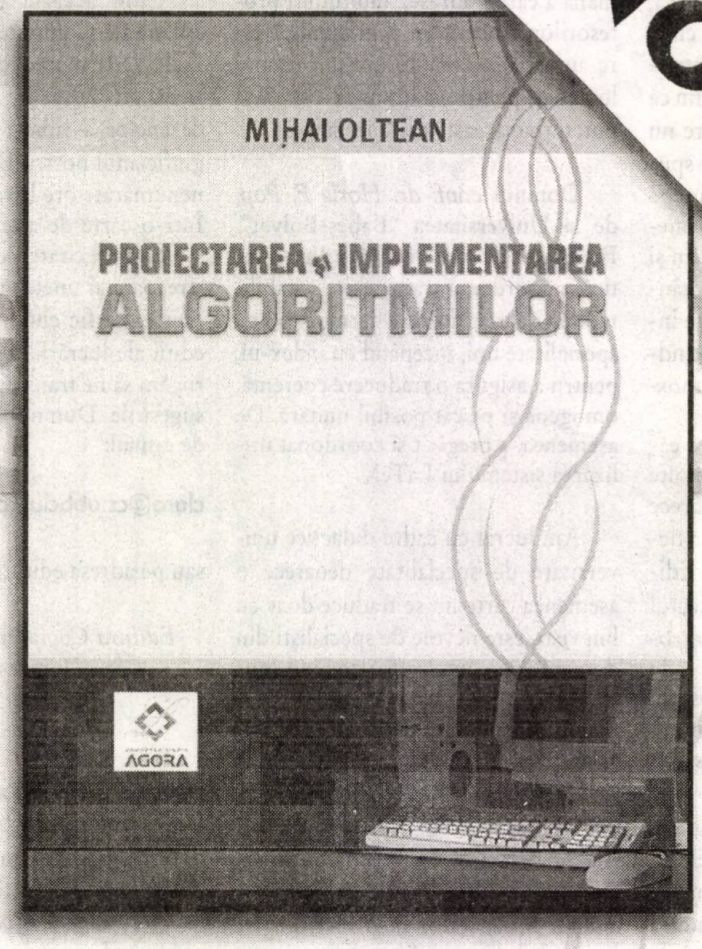
Autorii, așa cum am aflat de la MIT Press, lucrează la o versiune nouă a cărții, revizuită și îmbogățită. Vă promitem că, imediat ce această versiune va fi disponibilă pentru traducere, o vom publica și în limba română.

Clara Ionescu



COMPUTER LIBRIS AGORA

NOUTATE



Culegerea constituie un instrumentar pentru cei care doresc să intre în posesia secretelor proiectării algoritmilor. Va fi util când vă pregătiți pentru teză, pentru examenul de bacalaureat sau pentru admitere. Veți găsi în culegere și probleme care pot fi analizate și rezolvate în cercurile de elevi și care vor fi utile în pregătirea pentru concursurile școlare.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

CONCURS

A luat sfârșit și a patra etapă a concursului organizat de Gazeta de Informatică; în acest număr vă prezentăm rezolvările problemelor de la această rundă, clasamentul etapei, precum și situația după aceste patru runde. De asemenea, veți găsi și enunțurile celor trei probleme ale ultimei runde.

Ginfo Top 20

1.	Adrian Cârcu, Bistrița	1028
2.	Mihai Pătrașcu, Craiova	1006
3.	Cristian Cadar, București	975
4.	Silviu Bota, Baia Mare	962
5.	Daniel Dumitran, București	868
6.	Ioana Ileană, Alba Iulia	858
7.	Mihai Stroe, București	781
8.	Ciprian Baicu, Tn.-Severin	776
9.	Gabriel Erzse, Oradea	743
10.	Csaba Patcas, Oradea	703
11.	David Andrei, Rm. Vâlcea	693
12.	Vlad Vâlceanu, Arad	684
13.	Marian Anghel, Buftea	666
14.	Ileana Constantinescu, Sibiu	639
15.	Gabriel Pârvan, Rm. Sărat	622
16.	Robert Baban, Baia Mare	607
17.	Cosmin Negruseri, Bistrița	602
18.	Victor Asavei, Tn.-Severin	595
19.	Cristian Alexandrescu, Botoșani	544
20.	Sorin Otescu, Brașov	540

Termen:

Rezolvările problemelor publicate în acest număr pot fi trimise până la data de

30 Aprilie 2000

Runda 4 Top 10

1.	Silviu Bota, Baia Mare	297
1.	Gabriel Pârvan, Rm. Sărat	297
3.	Mihai Pătrașcu, Craiova	235
4.	Robert Baban, Baia Mare	215
5.	Cristian Cadar, București	212
5.	Mihai Stroe, București	212
5.	David Andrei, Rm. Vâlcea	212
5.	Marian Anghel, Buftea	212
5.	Ionuț Andreica, ***	212
10.	Ciprian Baicu, Tn.-Severin	206

Runda 4, problema 1

Labirint (Tom&Jerry)

Aceasta a fost problema cu punctajul mediu cel mai redus de până acum: 19.77 puncte. Aceasta s-a datorat, în mare măsură, faptului că mulți concurenți nu au înțeles că pozițiile lui Tom și Jerry sunt date în formatul *coloană, linie*. Noi am considerat că este destul de evident că x este coordonata orizontală și y cea verticală, dar se pare că nu am reușit decât să creăm confuzie, motiv pentru care ne cerem scuze.

Unii concurenți au rezolvat corect problema, dar nu au luat în considerare faptul că numărul drumurilor posibile poate avea peste 50 de cifre, deci trebuiau simulate operațiile cu numere mari; singurul care a rezolvat perfect problema a fost *Mihai Pătrașcu* din Craiova.

Problema era de fapt foarte simplă, drumul minim într-un labirint fiind foarte ușor de determinat, folosind, de exemplu, metoda programării dinamice (algoritmul lui Lee). Problema determinării numărului drumurilor minime nu este nici ea foarte dificilă; rezolvarea ei a fost prezentată în numărul 4/1999 al GInfo, problema *Drumuri pe autostradă* (enunțul apare în GInfo 2/1999). Vă prezentăm în continuare soluția propusă de Mihai Pătrașcu (poate înțelegeți ceva...).

Listing LABIRINT.CPP

```
#pragma option -3 -r -Z -O2 -a
#include <stdio.h>
#include <alloc.h>
#include <mem.h>
#define MSK_N       1
#define MSK_E       2
#define MSK_S       4
#define MSK_W       8
#define MSK_1       16
#define MSK_2       32
#define NUM_SIZE    32
#define ADD(i)\
  if((p[i]+=num[i])>=1000000000L)\
    p[i]-=1000000000L,p[(i)+1]++
#define TEST(wall,delta)\
  if(*p&(wall)){\
    if(!(p[delta]&(MSK_1|MSK_2))){\
      *(*(top++)=p delta)|=msk1;\
      copy_num(p,delta);\
    }\
    else if(!(p[delta]&msk2))\
      add_num(p,delta);\
    }

int n,m,i1,j1,i2,j2;
unsigned char neigh[100][100];
unsigned long huge res[10000][8],
                          far*num;

void read_in(void){
  int i,j,tmp;
  freopen("TOMJERRY.IN","r",stdin);
  scanf("%d%d%d%d%d%d",&n,&m,
                    &j1,&i1,&j2,&i2);
  i1--,j1--,i2--,j2--;
  for(i=0;i<n;i++)
    for(j=0;j<m;neigh[i][j++]=tmp)
      scanf("%d",&tmp);
}

void wr_num(void){
  int i=7;
  while(!num[--i]);
  for(printf("%lu",num[i]);i;
          printf("%09lu",num[--i]));
  putchar('\n');
}

void _fastcall add_num(int delta){
  unsigned long far*p =
                (unsigned long far*)\
                  ((char huge *)\
                    num+delta);
```

CR4P1 Top 5

1.	Mihai Pătrașcu, Craiova	100
2.	Silviu Bota, Baia Mare	97
2.	Gabriel Pârvan, Râmnicu Sărat	97
4.	Sorin Otescu, Brașov	94
5.	Ileana Constantinescu, Sibiu	91

```
  ADD(0);
  ADD(1);
  ADD(2);
  ADD(3);
  ADD(4);
  ADD(5);
  p[6]+=num[6];
}

void solve(void){
  unsigned char *q[10000],**bot,
                **top,**brk,*p,*target;
  int msk=MSK_1,mvs=0;
  *res[(*q=neigh[i1]+j1)-*neigh]=1;
  target=neigh[i2]+j2;
  for(top=(bot=q)+1;;msk^=MSK_1|MSK_2){
    for(++mvs,brk=top;bot<brk;
                        *p=MSK_1|MSK_2){
      num=(unsigned long far*)\
              res[(p=*(bot++))-*neigh];
      if(p==target){
        printf("%d\n", mvs);
        wr_num();
        return;
      }
      if(!(*p&MSK_S)){
        if(!(p[100]&(MSK_1|MSK_2))){
          *(*(top++)=p+100)|=msk;
          _fmemcpy((char huge*)num+
            100*NUM_SIZE,num,NUM_SIZE);
        }
        else
          if(!(p[100]&
                (msk^(MSK_1|MSK_2))))
            add_num(100*NUM_SIZE);
      }
      if(!(*p&MSK_N)){
        if(!(p[-100]&(MSK_1|MSK_2))){
          *(*(top++)=p-100)|=msk;
          _fmemcpy((char huge*)num-
            100*NUM_SIZE,num,NUM_SIZE);
        }
        else
          if(!(p[-100]&
                (msk^(MSK_1|MSK_2))))
            add_num(-100*NUM_SIZE);
      }
      if(!(*p&MSK_E)){
        if(!(p[1]&(MSK_1|MSK_2))){
          *(*(top++)=p+1)|=msk;
          _fmemcpy((char huge*)
            num+NUM_SIZE,num,NUM_SIZE);
        }
        else
          if(!(p[1]&
                (msk^(MSK_1|MSK_2))))
            add_num(NUM_SIZE);
      }
      if(!(*p&MSK_W)){
        if(!(p[-1]&(MSK_1|MSK_2))){
          *(*(top++)=p-1)|=msk;
          _fmemcpy((char huge*)
            num-NUM_SIZE,num,NUM_SIZE);
        }
        else
          if(!(p[-1]&
                (msk^(MSK_1|MSK_2))))
            add_num(-NUM_SIZE);
      }
    }
  }
}

int main(){
  read_in();
  freopen("TOMJERRY.OUT","w",stdout);
  solve();
}
```

Lizibilitate...

Această sursă ar trebui să primească premiul "GInfo Best Encrypted Solution!" Din cauza programatorilor care redactează codul astfel încât nimeni să nu poată descifra ceva, se spune că limbajul C nu poate fi folosit pentru învățarea algoritmicii, fiind mult prea complicat. În realitate aceste surse sunt complicate artificial, poate din dorința de a impresiona.

Nu putem decât să sperăm că asemenea tehnici copilărești de complicare inutilă a codului nu vor mai fi folosite de către tinerii programatori. Un astfel de programator nu se va putea integra într-o echipă, deoarece pentru o colaborare eficientă este nevoie, în primul rând, de comunicare. Dacă doriți doar să arătați cât de "șmecheri" sunteți, luați exemplu! Totuși vă rugăm să fiți atenți să nu faceți gafe care să stârnească zâmbetele celor care vă citesc sursele. De exemplu, în sursa anterioară, deși se definește macro-ul TEST, acesta nu este folosit niciodată. Sau poate a fost doar o altă strategie de complicare a sursei...

Runda 4, Problema 2

Star Trek (GInfo Generation)

Chiar dacă această problemă a fost destul de dificilă, numărul redus al concurenților care au încercat să o rezolve a dus la un punctaj mediu acceptabil, de exact 50 de puncte.

Rezolvarea problemei necesita cunoașterea a doi algoritmi fundamentali de geometrie computațională și anume determinarea înfășurătorii convexe a unei mulțimi de puncte (prezentat în GInfo 8/1998) și determinarea intersecției a două poligoane convexe (prezentat în GInfo 4-5/1998 și GInfo 8/1999). Aria poligonului intersecție este apoi foarte ușor de calculat, determinând o triangulație a poligonului și însumând aria triunghiurilor formate.

Vă prezentăm acum soluția propusă de *Gabriel Pârvan* din Râmnicu Sărat.

Listing STARTREK.PAS

```
type punct=record x,y:Extended end;
     polig=array[1..200]of punct;
     latura=record a,b:punct end;

var r,k,c:polig;
    n,m,nc,i,j:Integer;
    a:Extended; f:Text;
    z:array[1..100] of Byte;

procedure intersect(s,p:punct; x:latura;
                    var i:punct);
var p1,p2,p3,p4,p5,p6,p7,v:Extended;
begin
  with x do
  begin
    p1:=b.x-a.x; p2:=s.y-p.y;
    p3:=s.x-p.x; p4:=b.y-a.y;
    p5:=p1*p2-p3*p4; p6:=s.x-a.x;
    p7:=s.y-a.y; v:=(p1*p7-p6*p4)/p5;
    i.x:=s.x+(p.x-s.x)*v;
    i.y:=s.y+(p.y-s.y)*v
  end
end;

procedure decupare_latura(var vp:polig;
     np:Integer; x:latura; var l:polig;
                 var nl:Integer);
var j:Integer;
    s,p,i:punct;
    a,b,c:Extended;
begin
  nl:=0; s:=vp[np];
```

Au rezolvat corect:

David Andrei, Râmnicu Vâlcea
Mugurel Ionuț Andreica, ***
Marian Anghel, Buftea
Victor Asavei, Dr. Turnu-Severin
Robert Baban, Baia Mare
Ciprian Baicu, Dr. Turnu-Severin
Silviu Bota, Baia Mare
Cristian Cadar, București
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Horea-Adrian Haitonic, Năsăud
Ioana Ileană, Alba Iulia
Marius Muja, Orăștie
Sorin Otescu, Brașov
Gabriel Pârvan, Râmnicu Sărat
Mihai Stroe, București

```
  a:=x.b.y-x.a.y; b:=x.a.x-x.b.x;
  c:=x.a.y*x.b.x-x.a.x*x.b.y;
  for j:=1 to np do
  begin
    p:=vp[j];
    if a*p.x+b*p.y+c<=0 then
      if a*s.x+b*s.y+c<=0 then
        begin Inc(nl); l[nl]:=p end
      else
        begin
          intersect(s,p,x,i);
          Inc(nl); l[nl]:=i;
          Inc(nl); l[nl]:=p
        end
    else
      if a*s.x+b*s.y+c<=0 then
      begin
        intersect(s,p,x,i);
        Inc(nl);
        l[nl]:=i
      end;
    s:=p
  end
end;

procedure sutherland_hodgman(
        var p,q,l:polig; np,nq:Integer;
                         var nl:Integer);
var x:latura;
    j:Integer;
begin
  x.a:=p[np];
  for j:=1 to np do
  begin
    x.b:=p[j];
    decupare_latura(q,nq,x,l,nl);
    q:=l; nq:=nl; x.a:=x.b
  end
end;

procedure convex(var p:polig;
                         var np:integer);
var i,j,k:Integer; a,b,c:Extended;
    r:polig; s:set of 1..100;
begin
  j:=1;
  for i:=2 to np do
    if p[i].x<p[j].x then j:=i;
  z[1]:=j; k:=1; s:=[];
  repeat
    i:=1;
    while (i in s)or(i=z[k]) do Inc(i);
    a:=p[i].y-p[z[k]].y;
    b:=p[z[k]].x-p[i].x;
    c:=p[z[k]].y*p[i].x-p[z[k]].x*p[i].y;
    for j:=1 to np do
     if not(j in s)and(j<>i)and(j<>z[k])
        and(a*p[j].x+b*p[j].y+c>0) then
      begin
        i:=j; a:=p[i].y-p[z[k]].y;
        b:=p[z[k]].x-p[i].x;
        c:=p[z[k]].y*p[i].x-
                    p[z[k]].x*p[i].y
      end;
      Inc(k);
      z[k]:=i;
      Include(s,i)
  until i=z[1];
  r:=p;
  for i:=1 to k-1 do p[i]:=r[z[i]];
  np:=k-1
end;

function area(p:polig; np:Integer):
                                Extended;
  function a3(a,b,c:punct):Extended;
  begin
    a3:=abs((b.x-a.x)*(c.y-a.y)-
                  (c.x-a.x)*(b.y-a.y))/2
  end;

var t:Extended; i:integer;
begin
  t:=0;
  for i:=3 to np do
    t:=t+a3(p[1],p[i-1],p[i]);
  area:=t
end;

Begin
  Assign(f,'STARTREK.IN'); Reset(f);
  Readln(f,m);
  for i:=1 to m do
    Readln(f,r[i].x,r[i].y);
  Readln(f,n);
  for i:=1 to n do
    Readln(f,k[i].x,k[i].y);
  Close(f);
  if (n<3)or(m<3) then a:=0
  else
    begin
      convex(r,m);
      convex(k,n);
      sutherland_hodgman(r,k,c,m,n,nc);
      a:=area(c,nc)
    end;
  Assign(f,'STARTREK.OUT'); Rewrite(f);
  Writeln(f,a:0:4);
  Close(f)
End.
```

Runda 4, problema 3

Ordonare
(algoritm la nivel de biți)

Aceasta a fost o problemă foarte ușoară; ea a necesitat doar cunoașterea modului de codificare a numerelor distincte folosind algoritmi pe biți (prezentați în numărul 7/1999 al GInfo); punctajul mediu obținut a fost de 50.23 puncte.

Avem 1000000 de biți (125000 bytes) pentru a codifica numerele din intervalul [0,999999]. Dacă un anumit număr apare în fișierul de intrare, vom atribui valoarea 1 bitului corespunzător.

Pentru afișarea numerelor în ordine crescătoare vom parcurge șirul de biți (începând de la 0) și, în cazul în care un bit are valoarea 1, vom scrie numărul corespunzător în fișierul de ieșire.

Vă prezentăm în continuare soluția propusă de *Dumitran Daniel* din București.

Listing NUMERE.CPP

```
#include <stdio.h>
#include <mem.h>
#define lenbuff 125000
#define fisin "NUMERE.IN"
#define fisout "NUMERE.OUT"

FILE *f;
char huge buff[lenbuff];
unsigned long nr,i,j;

void main(){
  _fmemset(buff,0,lenbuff);
  f=fopen(fisin,"r");
  while(!feof(f)){
    fscanf(f,"%lu",&nr);
    buff[nr>>3]|=1<<(nr&7);
  }
  fclose(f);
  f=fopen(fisout,"w");
  for(i=0l;i<lenbuff;i++){
    if(!buff[i]) continue;
    for(j=0l;j<8l;j++)
      if(buff[i]&(1<<j))
        fprintf(f,"%lu\n",(i<<3)|j);
  }
  fclose(f);
}
```

Au rezolvat corect:

David Andrei, Râmnicu Vâlcea

Mugurel Ionuț Andreica, ***

Liviu Andreicuț, București

Marian Anghel, Buftea

Victor Asavei, Dr. Turnu-Severin

Ciprian Baicu, Dr. Turnu-Severin

Silviu Bota, Baia Mare

Cristian Cadar, București

Adrian Cârcu, Bistrița

Ileana Constantinescu, Sibiu

Daniel Dumitran, București

Gabriel Pârvan, Râmnicu Sărat

Csaba Patcas, Oradea

Mihai Pătrașcu, Craiova

Liviu Păunescu, Constanța

Daniel Popa, Orăștie

Mihai Stroe, București

Ionuț Trifan, Buzău

Runda 6

CR6P1: Galbeni

Se dau trei grămezi, fiecare conținând un număr de galbeni! La fiecare pas poate fi efectuată operația descrisă în continuare. Se aleg două dintre cele trei grămezi. Să presupunem că prima grămadă aleasă este A și ea conține x galbeni, iar a doua este B și conține y galbeni. Din grămada A se vor muta, în grămada B, y galbeni; după operație, grămada A va conține x-y galbeni, iar grămada B va conține $2y$ galbeni. Scrieți un program care determină, în mai puțin de 10 secunde, o secvență de operații care trebuie efectuate astfel încât să obținem două grămezi care conțin același număr de galbeni. Fiecare grămadă conține cel puțin un galben, iar numărul total al galbenilor nu poate depăși 4000000000 (patru miliarde). Datele de intrare vor fi citite din fișierul **GOLD.IN** care va conține trei numere separate printr-un singur spațiu, reprezentând numărul de galbeni din fiecare dintre cele trei grămezi. În fișierul de ieșire **GOLD.OUT** se va scrie, pe câte o linie, fiecare operație necesară pentru obținerea în final a două grămezi care conțin un număr egal de galbeni. Pentru descrierea unei operații veți folosi două numere separate printr-un spațiu. Primul număr va identifica grămada din care se vor lua galbeni, iar al doilea grămada în care se vor pune galbeni. Valorile celor două numere pot fi 1, 2 sau 3. Nu este obligatoriu ca programul să furnizeze o soluție cu număr minim de pași.

Exemplu

GOLD.IN

```
3703703670 1234575261 2457
```

GOLD.OUT

```
2 3
2 3
1 2
```

Explicație

La primul pas vom muta 2457 de galbeni din grămada 2 în grămada 3. Grămezile vor conține 3703703670, 1234572804, respectiv 4914 de galbeni. La al doilea pas vom muta 4914 galbeni din grămada 2 în grămada 3. Grămezile vor conține 3703703670, 1234567890, respectiv 9828 galbeni. La ultimul pas vom muta 1234567890 galbeni din grămada 1 în grămada 2. Grămezile vor conține 2469135780, 2469135780, respectiv 9828 de galbeni, deci primele două grămezi conțin același număr de galbeni.

CR6P2: Rubik

Se dă o grilă de dimensiuni $m \times n$ (m linii, n coloane) care conține în fiecare celulă, un număr natural din intervalul [1,1000]. O operație constă în a aduna valoarea k ($-1000 \leq k \leq 1000$) la numerele conținute într-o anumită celulă și în toate celulele adiacente pe verticală sau orizontală (nu și pe diagonală). Determinați o secvență de operații care duce la obținerea aceleiași valori v în toate cele $m \times n$ celule ale grilei. Datele de intrare sunt citite din fișierul **RUBIK.IN** care conține pe prima linie numerele m și n, separate printr-un singur spațiu, iar pe următoarele m linii câte n numere. Al j-lea număr de pe linia i a fișierului va corespunde numărului din a j-a celulă de pe linia i a grilei. Numerele de pe aceeași linie vor fi separate prin spații. Datele de ieșire vor fi scrise în fișierul **RUBIK.OUT**. Prima linie va conține valoarea v, iar a doua numărul x al operațiilor care trebuie efectuate. Următoarele x linii vor conține trei numere. Primele două vor indica linia și coloana (în această ordine) celulei alese pentru operație, iar al treilea valoarea k care va fi adunată la numărul din celulă și din celulele adiacente pe orizontală sau verticală. Programul va trebui să furnizeze, în cel mult 10 secunde, o secvență corectă de operații, nu neapărat cu număr minim de pași.

Exemplu

RUBIK.IN

```
5 5
21 26 20 21 21
26 26 25 20 21
22 27 21 17 21
10 22 17 17 17
10 10 21 17 21
```

RUBIK.OUT

```
21
5
2 2 -5
2 3 1
3 2 -1
4 4 4
5 1 11
```

CR6P3: Expresie

Să se construiască o expresie aritmetică parantezată de valoare apropiată de o valoare dată. Expresia poate conține doar operanzii furnizați la intrare. Nu e neapărat nevoie să fie folosiți toți operanzii, dar fiecare poate fi folosit de mai multe ori. Se folosesc doar operatorii *, + și sunt permise și paranteze. Datele de intrare se citesc din fișierul **INPUT.TXT** care conține pe prima linie valoarea e ($1 \leq e \leq 2000000000$) a expresiei. Pe a doua linie se dau operanzii permiși, cel mult 10, separați prin câte un spațiu. Expresia va fi scrisă în fișierul **OUTPUT.TXT** și va conține maximum 100 de operatori.

Exemplu

INPUT.TXT

```
50
3 6 10 2
```

OUTPUT.TXT

```
10*(3+2)
```

Observație

Se acordă punctaje parțiale.

BACALAUREAT 2000

Continuăm prezentarea soluțiilor subiectelor de Bacalaureat propuse spre rezolvare în cartea Bac Info' 2000; în acest număr veți găsi varianta nr. 18.

Subiectul 1

1. Scrieți o secvență de instrucțiuni în pseudocod, care să aibă ca efect determinarea numărului cifrelor numărului natural n.

Soluție

Deoarece numărul poate fi și 0, algoritmul trebuie să afișeze 1 în acest caz. Altfel, acesta se împarte la 10 cât timp este diferit de 0. La fiecare împărțire, contorul crește cu 1.

```
citeşte n
dacă n=0 atunci scrie '1'
          altfel
            nr_cifre ← 0
            cât timp n > 0 execută
              n ← [n/10]
              nr_cifre ← nr_cifre+1
            sfârşit cât timp
            scrie nr_cifre
sfârşit dacă
```

2. Care dintre următoarele afirmații sunt adevărate și care sunt false?

a) Orice algoritm poate fi scris, folosind doar structuri secvențiale, alternative de tip *if-then-else* și structuri repetitive anterior condiționate.

b) Într-un program structurat nu se va folosi instrucțiunea **goto**.

c) Utilizarea instrucțiunii **case** în (*Pascal*), respectiv **switch** (în *C*) contravine principiilor programării structurate.

d) Conform principiilor programării structurate nu se vor efectua salturi forțate din interiorul unei structuri repetitive.

e) În cazul folosirii instrucțiunii **goto**, în program trebuie să existe în mod obligatoriu o instrucțiune etichetată cu eticheta din **goto**.

Justificați răspunsurile.

Soluție

a) Afirmația este adevărată, deoarece exprimă exact *teorema lui Böhm și Jacopini*.

b) Afirmația este adevărată, deoarece principiile programării structurate interzic folosire instrucțiunii **goto** în programe.

c) Afirmația este falsă, deoarece utilizarea celor două instrucțiuni (**case** în *Pascal*, respectiv **switch** în *C*) nu contravine principiilor programării structurate. Aceste instrucțiuni sunt similare cu instrucțiunea **if**, dar realizează o ramificare multiplă.

d) Afirmația este adevărată; conform principiilor programării structurate nu se vor efectua salturi forțate din interiorul unei structuri repetitive.

e) Afirmația este adevărată, deoarece folosirea instrucțiunii **goto** înseamnă solicitarea continuării execuției unui program nu cu instrucțiunea imediat următoare, ci cu o instrucțiune aflată oriunde în blocul respectiv, care trebuie să fie identificabilă. Identificarea ei se realizează prin intermediul etichetei.

3. Care este efectul următorului subprogram?

Varianta Pascal:

```
procedure oare_ce_e(var _text:string;
                                n,k:Byte);
var i,j:Byte;
    aux:Char;
begin
  for j:=1 to k do
  begin
    aux:=_text[1];
    for i:=2 to n do
      _text[i-1]:=_text[i];
    _text[n]:=aux
  end
end;
```

Varianta C:

```
void oare_ce_e(char* _text,int n,int k){
char aux;
for (int j=0;j<k;j++){
  aux=_text[0];
  for (int i=1;i<n;i++)
    _text[i-1]=_text[i];
  _text[n-1]=aux;
  }
}
```

Soluție

Procedura (funcția) oare_ce_e realizează o permutare circulară a șirului de caractere _text cu k poziții.

4. Ce se va afișa pe ecran în urma execuției programului:

Varianta Pascal:

```
Program Nu_e_ghicitoare;
var x,y,u,v:Byte;
Begin
  Readln(x,y); u:=x; v:=y;
  while x<>y do
    if x>y then begin x:=x-y; u:=u+v end
    else begin y:=y-x; v:=v+u end;
  Writeln(x,' ',(u+v) div 2)
End.
```

Varianta C:

```
#include <stdio.h>
void main(void){
  int x,y,u,v;
  scanf("%d%d",&x,&y);
  while(x!=y)
    if(x>y)x-=y,u+=v;
    else y-=x,v+=u;
  printf("%d %d",x,(u+v)/2);
}
```

Soluție

Programul calculează (fără să efectueze nici o înmulțire și doar o împărțire la 2) cel mai mare divizor comun, respectiv cel mai mic multiplu comun a două numere x și y.

Subiectul 2

1. Scrieți un program comentat care convertește un număr dat în baza p în baza q și afișează această reprezentare (p,q≤10).

Soluție

Vom converti în baza 10 numărul dat (dacă acesta nu este dat în baza 10) și apoi numărul obținut îl vom vconverti în baza q (dacă nu trebuie convertit în baza 10). Dacă p=q atunci nu are rost să convertim numărul.

Varianta Pascal:

Listing CONVERSIE.PAS

```
Program Conversie;
var n,nr_nou,p,q,cifra,putere:Longint;
Begin
  Write('n='); Readln(n);
  Write('p= (<=10)'); Readln(p);
  Write('q= (<=10)'); Readln(q);
  if p=q
  then
  begin
    Writeln('p=q'); Halt
  end;
  if p<>10 then
  begin
    nr_nou:=0;
    putere:=1;
    while n>0 do
    begin
      cifra:=n mod 10;
      n:=n div 10;
      nr_nou:=nr_nou+putere*cifra;
      putere:=putere*p
    end;
    n:=nr_nou
  end;
  if q<>10 then
  begin
    nr_nou:=0;
    putere:=1;
    while n>0 do
    begin
      cifra:=n mod q;
      n:=n div q;
      nr_nou:=nr_nou+putere*cifra;
      putere:=putere*10
    end;
    n:=nr_nou
  end;
  Writeln('Numarul convertit: ',n);
End.
```

Varianta C:

Listing CONVERSIE.CPP

```
#include <stdio.h>

void main(void){
  long n,nr_nou,p,q,cifra,putere;
  printf("n="); scanf("%ld",&n);
  printf("p=(<=10):"); scanf("%ld",&p);
  printf("q=(<=10):"); scanf("%ld",&q);
  if (p==q){
    puts("p=q");
    return;
  }
  if (p!=10){
    nr_nou=0,putere=1;
    while(n){
      cifra=n%10,n/=10;
      nr_nou+=putere*cifra;
      putere*=p;
    }
    n=nr_nou;
  }
  if (q!=10){
    nr_nou=0;
    putere=1;
```

```
    while(n){
      cifra=n%q,n/=q;
      nr_nou+=putere*cifra;
      putere*=10;
    }
    n=nr_nou;
  }
  printf("Numarul convertit %ld.\n",n);
}
```

2. Scrieți un program recursiv care citește n cuvinte de la tastatură și le afișează în ordinea inversă citirii. (Cuvintele nu vor fi memorate într-un tablou.)

Soluție

Varianta Pascal:

Listing CUVINTE.PAS

```
Program cuvinte;
var n:Byte;

procedure inversare(i:Byte);
var cuvant:string;
begin
  Readln(cuvant);
  if i<n
  then inversare(i+1)
  else Writeln('Cuvintele invers');
  Writeln(cuvant)
end;

Begin
  Write('n='); Readln(n);
  Writeln('Cuvintele: ');
  Inversare(1)
End.
```

Varianta C:

Listing CUVINTE.CPP

```
#include <stdio.h>

void inversare(int i){
  char cuvant[100];
  fscanf("%s",cuvant);
  if (i<n) inversare(i+1);
  else puts("Cuvintele invers");
  puts(cuvant);
}

void main(void){
  int n;
  printf("n="); scanf("%d",&n);
  inversare(1);
}
```

Subiectul 3

Se citește de la tastatură un număr natural nenul n≤8. Să se genereze, folosind metoda *backtracking*, toate șirurile de lungime n, formate din cifre de 0 și 1. Programul va fi comentat.

Exemplu: pentru n=2 se va afișa:

```
0 0
0 1
1 0
1 1
```

Soluție

Varianta Pascal:

Listing _0_1.PAS

```
Program _0_1;
var n:Byte;
    sir:array[1..8] of 0..1;

procedure scrie;
var j:Byte;
begin
  for j:=1 to n do Write(sir[j]);
  Writeln
end;

procedure generare(i:Byte);
var j:Byte; { i = indicele elementului }
            {          curent din sir }
begin
  for j:=0 to 1 do { valorile lui j  }
  begin    { sunt elementele posibile }
    sir[i]:=j;   { ale sirului cerut }
    if i<n then generare(i+1)
    else scrie      { avem n elemente }
  end               {  se poate scrie }
end;                {  un sir generat }
Begin
  Write('n='); Readln(n);
  generare(1)
End.
```

Varianta C:

Listing _0_1.CPP

```
#include <stdio.h>

int n,sir[8];

void scrie(void){
  for(int j=0;j<n;j++)
    printf("%d",sir[j]);
  putchar('\n');
}
```

```
void generare(int i){
      // i=indicele elementului curent
  for(int j=0;j<2;j++){
        // j=elementele posibile ale
                    // sirului cerut
    sir[i]=j;
    if(i<n-1) generare(i+1);
    else scrie(); // avem n elemete
  }               // se poate scrie
}                 // un sir generat

void main(void){
  printf("n="); scanf("%d",&n);
  generare(0);
}
```

Subiectul 4

1. Un fișier text al cărui nume se introduce de la tastatură cuprinde numele și media de la examenul de bacalaureat a unor absolvenți. Scrieți programul comentat care calculează media generală a absolvenților din fișier și scrie această medie într-un alt fișier.

Soluție

Varianta Pascal:

Listing BAC.PAS

```
Program Bac;
type articol=record
                nume:string[20];
                medie:Real
              end;
var s,med_gen:Real;
    f,g:Text;
    fis_in,fis_out:string;
    absolvent:articol;
    n:Byte;
    car:Char;
Begin
  Writeln('Fisier de intrare: ');
  Readln(fis_in);
  Writeln('Fisierul de iesire: ');
  Readln(fis_out);
  Assign(f,fis_in); Reset(f);
  Assign(g,fis_out); Rewrite(g);
  s:=0;
  while not Eof(f) do
  begin
    Inc(n); { numaram absolventii }
    with absolvent do
    begin
      nume:='';
      Read(f,car); { primul caracter }
      while car<>' ' do
                { caracterele numelui }
      begin
        nume:=nume+car;
        Read(f,car)
      end;
      Readln(f,medie); { citim media }
      s:=s+medie      { calculam suma }
    end
  end;
  if n<>0           { avem absolventi }
  then
  begin
    med_gen:=s/n;    { media generala }
    Writeln(g,med_gen:12:2)
  end
  else
    Writeln(g,'Nu sunt absolventi.');
  Close(f);
  Close(g)
End.
```

Varianta C:

Listing BAC.CPP

```
#include <stdio.h>
void main(void){
  char fis_in[20],fis_out[20];
  printf("Fisierul de intrare:");
  scanf("%s",fis_in);
  FILE *f=fopen(fis_in,"rt");
  printf("Fisierul de iesire:");
  scanf("%s",fis_out);
  FILE *g=fopen(fis_out,"wt");
  float s=0; int n=0;
  while (!feof(f)){
    n++; // numaram absolventii
    struct articol{
      char nume[20];
      float medie;
    }absolvent;
    fscanf(f,"%s%f\n",absolvent.nume,
                      &absolvent.medie);
    s+=absolvent.medie; // suma
  }
  if (n){
    float med_gen=s/n; // media gen.
    fprintf(g,"%12.2f",med_gen);
  }
  else
    fputs("Nu sunt absolventi.",g);
  fclose(f);
  fclose(g);
}
```

2. Scrieți două subprograme care realizează inserarea, respectiv eliminarea unui element într-o coadă. Coada a fost definită astfel:

Varianta Pascal:

```
type coada=^data;
     data=record
            an:1900..2000;
            luna:1..12;
            zi:1..31;
            urm:coada
          end;
```

Varianta C:

```
struct coada{
  int an,luna,zi;
  coada* urm;
};
```

Subprogramul va avea doi parametri de tip coada care indică primul, respectiv ultimul element existent în coadă.

Soluție

Pentru simplificare considerăm că am folosit o implementare cu santinelă a cozii. Inserarea în coadă se realizează după ultimul element, iar eliminarea din coadă înseamnă ștergerea elementului care urmează după santinelă, deci, de fapt, primul element "real" al cozii. Evident, înainte de a elimina un element va trebui să verificăm dacă există cel puțin un element în coadă. De obicei se dorește să fie eliberat spațiul alocat elementului care se elimină. Presupunând că cei doi pointeri definiți și inițializați, către primul (santinela), respectiv, ultimul element din coadă sunt primul și ultimul, inserarea și eliminarea se realizează cu următoarele subprograme:

Varianta Pascal:

```
procedure inserare(var ultimul:coada);
var p:coada;
begin
  New(p);
  Readln(p^.an,p^.luna,p^.zi);
  p^.urm:=nil
  ultimul^.urm:=p;
  ultimul:=p
end;

procedure eliminare(var primul:coada);
var p:coada;
begin
  if primul^.urm = nil
  then Writeln('Coada este vida!')
  else
  begin
    p:=primul^.urm;
    primul^.urm:=primul^.urm^.urm;
    Dispose(p)
  end
end;
```

Varianta C:

```
void inserare(coada* &ultimul){
  coada *p=new coada;
  fscanf("%d%d%d",p->an,p->luna,p->zi);
  p->urm=NULL;
  ultimul->urm=p;
  ultimul=p;
}

void eliminare(coada* &primul){
  coada *p=new coada;
  if (!primul->urm) puts("Coada este vida!");
  else{
    p=primul->urm;
    primul->urm=primul->urm->urm;
    delete p;
  }
}
```

Barem

Subiectul 1: 30 puncte
- 1) 7,5 puncte
- 2) 7,5 puncte
- 3) 7,5 puncte
- 4) 7,5 puncte

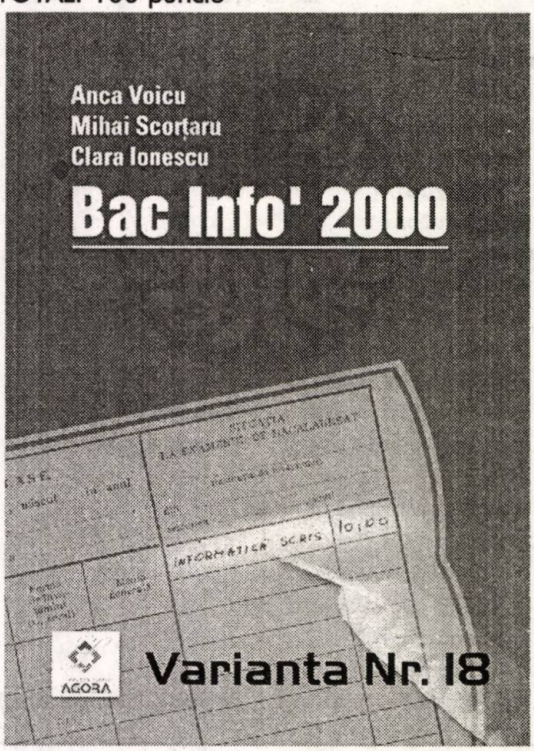
Subiectul 2: 20 puncte
- 1) 10 puncte
- 2) 10 puncte

Subiectul 3: 20 puncte

Subiectul 4: 20 puncte
- 1)10 puncte
- 2) 10 puncte

Se acordă 10 puncte din oficiu.

TOTAL: 100 puncte



Rolul DECIZIEI în programarea DINAMICĂ

Ilie Vieru

În literatura de specialitate programarea dinamică este recomandată pentru acele probleme pentru care se cere să se afle soluția optimă dintr-o mulțime finită de soluții posibile. Metodele euristice, de obicei, nu conduc la o soluție optimă, iar metoda backtracking este foarte lentă.

Principiile generale ale *programării dinamice* sunt foarte bine precizate în toate lucrările de referință, cunoscându-se trei strategii de rezolvare a problemelor folosind această metodă (*înainte*, *înapoi* și *mixtă*).

Dificultățile apar însă la alegerea uneia din cele trei strategii, la stabilirea relațiilor de recurență corespunzătoare formei în care este verificat principiul optimalității.

Etapa asupra căreia vom insista este *rolul deciziilor* în trecerea sistemului dintr-o stare în alta, dar mai ales, în valorificarea rezultatelor puse la dispoziție de metodă. În continuare vom arăta, prin câteva exemple cunoscute, că memorând deciziile în fiecare etapă (lucru care este ineficient doar din punct de vedere al consumului de memorie) putem alege atât strategia *înainte* cât și strategia *înapoi*, iar evidențierea unei soluții care conduce sistemul din starea inițială în cea finală în condiții de optim, este foarte simplă.

Cel mai lung subșir crescător

Se consideră un vector V cu n numere întregi. Se cere să se afișeze cel mai lung subșir crescător al acestuia.[1]

Soluție

Notăm cu $F(k)$ lungimea celui mai lung subșir crescător care se poate forma începând cu elementul k al vectorului și cu $d(k)$ decizia în poziția k, adică rangul termenului din șir care va urma în subșirul construit imediat după elementul $V(k)$. Obținem astfel relațiile următoare pentru strategia înainte:

$$F(k)=\begin{cases}1, \text{pentru } k=n\\ 1+\max\{F(i)\mid V(i)\ge V(k), i=\overline{k+1,n}\}, \text{ pentru } k<n\end{cases}$$

$$d(k)=\begin{cases}0, \text{pentru } k=n;\\ i, \text{unde } i \text{ reprezintă ordinul elementului pentru care}\\ \quad \text{s-a determinat valoarea lui } F(k), k<n;\\ \text{(dacă } F(k)=1, \text{atunci } d(k)=0).\end{cases}$$

Șirurile obținute ne permit să determinăm atât $max\{F(k) \mid k = 1..n\}$ cât și termenii care compun un astfel de subșir de dimensiune maximă, folosind valorile date de k, $d(k)$, $d(d(k))$, ...

Informațiile sunt suficiente ca să putem obține foarte ușor toate subșirurile de lungime maximă cu această proprietate.

Strategia *înapoi* presupune mici modificări de forma:

$$F(k)=\begin{cases}1, \text{ pentru } k=1;\\ 1+\max\{F(i)\mid V(i)\le V(k), i=\overline{1,k-1}\}, \text{ pentru } k>1\end{cases}$$

Triunghiul

Se consideră un triunghi de numere în care pe prima linie apare un număr, pe a doua linie apar două numere etc. Scrieți un program care calculează cea mai mare dintre sumele numerelor care apar pe drumurile care pleacă din vârf și ajung la bază, astfel încât în fiecare drum succesorul unui număr se află pe următorul rând, fie sub acesta, fie pe diagonală, la dreapta. Se citește numărul n al liniilor care formează triunghiul precum și elementele acestuia, numere întregi din intervalul [0, 99]. *(OII, Suedia, 1994)*

Soluție

Notăm cu a_{ij} numărul situat în triunghi pe linia i și coloana j, cu F_{ij} suma maximă a numerelor care apar pe drumurile care pleacă din a_{ij} și ajung la bază și cu d_{ij} poziția elementului din linia $i+1$ care urmează elementului a_{ij} pe drumul de sumă maximă.

În aceste condiții relațiile de recurență devin:

$$F_{ij}=\begin{cases}a_{ij}, \text{pentru } i=n;\\ a_{ij}+\max\{F_{i+1,j}, F_{i+1,j+1}\}, \text{pentru } i<n;\end{cases}$$

$$d_{ij}=\begin{cases}0, \text{pentru } i=n;\\ j \text{ sau } j+1, \text{după cum } F_{i+1,j}>F_{i+1,j+1}, \text{ sau}\\ \qquad F_{i+1,j}<F_{i+1,j+1}; \text{ pentru } i<n.\end{cases}$$

Folosind aceste relații și aplicând strategia *înainte* a *programării dinamice*, se determină F_{11} cât și drumul pe care se obține F_{11}.

Strategia *înapoi* poate fi folosită la fel de simplu, dacă vom considera F_{ij} ca fiind suma maximă a numerelor care apar pe drumurile care pleacă din vârful triunghiului (a_{11}) și ajung în a_{ij}. Atunci,

$$F_{ij}=\begin{cases} a_{11}, \text{pentru } i=1; \\ a_{ij}+\max\{F_{i-1,j}, F_{i-1,j-1}\}, \text{pentru } i>1; \end{cases}$$

$$d_{ij}=\begin{cases} 0, \text{pentru } i=1 \\ j \text{ sau } j+1, \text{după cum } F_{i+1,j}>F_{i+1,j+1}, \\ \quad \text{sau } F_{i+1,j}<F_{i+1,j+1}; \text{pentru } i>1. \end{cases}$$

Drumuri minime

Se consideră un graf orientat și matricea A a costurilor atașate arcelor grafului. Se cere ca, pentru fiecare pereche de vârfuri (i, j), să se tipărească lungimea drumului minim de la i la j.

În această problemă se aplică principiul optimalității în forma *mixtă*, în care funcția obiectiv are forma:

$$F_{ik}^{k}=\begin{cases} F_{ii}, \text{pentru } k=0 \\ \min\{F_{ij}^{k-1}, F_{ik}^{k-1}+F_{kj}^{k-1}\}, \text{pentru } k\neq 0 \end{cases}$$

și este asociată drumului de cost minim de la i la j, drum care poate trece numai prin vârfurile $1, 2, \ldots, k$.

Pentru fiecare etapă k, decizia are forma:

$$d(i,j)=\begin{cases} k, \text{ dacă } F_{ij}^{k}=F_{ik}^{k-1}+F_{kj}^{k-1} \\ d(i,j), \text{ în rest.} \end{cases}$$

Inițial avem $d(i, j) = 0$, $\forall i, j = \overline{1,n}$. Un rezultat interesant se poate formula astfel: *pentru oricare două vârfuri i, j din graf putem determina atât costul minim* F_{ij}^{n}, *cât și drumul de cost minim cu extremitățile în i, respectiv în j, folosind metoda divide et impera în matricea deciziilor, astfel:*

```
procedure afis(i,j:Integer);
begin
  if d(i,j)=0
    then
      Write(i,' ')
  else
    begin
      afis(i,d[i,j]);
      afis(d[i,j],j)
    end
end;
```

Nu vom încheia articolul înainte de a prezenta o problemă celebră care a preocupat mulți matematicieni și anume problema partițiilor unui număr întreg [3].

Partiționare

Orice număr întreg se poate descompune în termeni; astfel, numărul 4 se poate descompune în 5 moduri: 4=3+1=2+2=2+1+1=1+1+1+1. Dacă notăm cu n numărul de descompus și cu $p(n)$ numărul de moduri în care se poate descompune, putem scrie: $p(1)=1$, $p(2)=2$, $p(3)=3$, $p(4)=5$, $p(5)=7$, $p(6)=11$, etc. Să se determine $p(n)$, $\forall n\in \mathbf{N}$, $n>1$.

Problema a fost abordată de Euler, apoi de matematicianul MacMahon care a calculat $p(100)=190569292$. Mai târziu s-au calculat și alte valori pentru $p(n)$.

Pentru rezolvarea problemei vom introduce notația $p(n, k)$ reprezentând numărul de moduri în care se poate partiționa numărul n în așa fel încât fiecare partiție să conțină cel puțin un termen cu valoarea k ($1\le k\le n$), iar restul termenilor având valori mai mici sau egale cu k.

Convenim să reprezentăm prin $p(n, 0)$ numărul total al partițiilor lui n (adică $p(n)$).

De exemplu, avem $p(6, 1)=1$; $p(6, 2)=3$, $p(6, 3)=3$, $p(6, 4)=2$, $p(6, 5)=1$, $p(6, 6)=1$, iar $p(6, 0)=11$.

Un alt exemplu pe care-l apreciem ca suficient este $p(7, 1)=1$; $p(7, 2)=p(5, 2) + p(5, 1)$; $p(7, 3) = p(4, 3) + p(4, 2) + p(4, 1)$; $p(7, 4) = p(3, 0)$; $p(7, 5) = p(2, 0)$; $p(7, 6) = p(1, 0)$; $p(7, 7)=1$.

Vă propunem în continuare următorul algoritm de programare dinamică (vezi și [4]):

$$p(n,k)=\begin{cases} 1, \text{ pentru } k=n \\ p(n-k,0), \text{ pentru } n\le 2k \\ \sum_{i=1}^{k} p(n-k,i), \text{în rest.} \end{cases} \quad (1)$$

$$p(n,0)=\sum_{k=1}^{n} p(n,k) \quad (2)$$

unde $p(1, 0) = p(1, 1) = 1$. (3)

Folosind relațiile (1) și (2) și ținând cont de condițiile inițiale (3) programul se implementează ușor, obținând valoarea lui $p(n)$ într-un timp excelent, algoritmul având un ordin de complexitate polinomial.

Se observă și următoarea recurență:

$p(n, 1) = p(n, 2) = p(n, 3) + \ldots + p(n, k) = p(n+k, k)$ (4)

care se regăsește și în [3], unde $p(n, k)$ are o altă semnifi-

cație și anume $p(n, m)$ reprezintă numărul de partiții ale numărului n care sunt formate din m termeni:

$n=a_1+a_2+...+a_n$ cu proprietatea: $a_1 \geq a_2 \geq ... \geq a_n$.

Programul care implementează acest algoritm este prezentat în continuare.

Listing NUMERE.PAS

```
program Numere;
var p:array[1..100,0..100] of Longint;
    n,i,k,j,l:Integer;
Begin
  Writeln('Dati n');
  Readln(n);
  p[1,0]:=1;
  p[1,1]:=1;
  for i:=2 to n do
  begin
    p[i,0]:=0;
    for k:=1 to i do
    begin
      if k=i
        then
          p[i,k]:=1
      else
        if i<2*k
          then
            p[i,k]:=p[i-k,0]
        else
          begin
            p[i,k]:=p[i-k,k];
            for l:=1 to k-1 do
              p[i,k]:=p[i,k]+p[i-k,l]
          end;
      p[i,0]:=p[i,0]+p[i,k]
    end
  end;
  Writeln('p(',n,')=',p[n,0])
End.
```

Bibliografie

1. Tudor Sorin, *Tehnici de programare*, 1994
2. Doina Rancea, *Algoritmi fundamentali*, 1999.
3. Ioan Tomescu, *Combinatorica și teoria grafurilor*, 1978.
4. Ilie Vieru, *Câteva aspecte legate de metodica predării tehnicii de programare "Programarea dinamică" - Revista "Să înțelegem știința"*, nr. 2-3/1999.

Domnul Ilie Vieru este cadru didactic la Liceul "Gh.Vrânceanu" din Bacău și poate fi contactat prin e-mail la adresa dimex@cybernet.ro

news...

Borland C++Builder 5

Pe data de 22 Martie 2000 compania Inprise a lansat versiunea 5 a cunoscutului său mediu de dezvoltare vizuală C++Builder. Noua versiune este disponibilă în trei variante C++ Builder 5 Enterprise, C++Builder 5 Professional și C++Builder 5 Standard.

C++Builder 5 oferă suport pentru ultimele standarde Internet cum ar fi XML 1.0 și HTML 4.0 pentru a permite dezvoltarea de aplicații profesionale pentru servere Web.

Noii clienți ai firmei Inprise vor putea achiziționa Borland C++Builder 5 Enterprise la un preț de 2499$, Borland C++Builder 5 Professional la 799$ și Borland C++Builder 5 Standard la 99.95$. Actualii posesori ai unui produs Inprise din categoria client/server sau enterprise vor putea achiziționa Borland C++ Builder 5 Enterprise cu 1699$. Posesorii unui produs Inprise din categoria professional vor putea beneficia de facilitățile Borland C++ Builder 5 Enterprise la un preț de 2199$. Posesorii unui alt produs de dezvoltare Inprise pot achiziționa Borland C++Builder 5 Professional cu 249.95$.

Borland JBuilder 3.5

Pe data de 14 Martie 2000 compania Inprise a lansat versiunea 3.5 a mediului de dezvoltare vizuală JBuilder. Noua versiune este disponibilă în două variante JBuilder 3.5 Enterprise și JBuilder 3.5 Professional.

JBuilder 3.5 este integrat cu Inprise Application Server care oferă suport pentru standardele de tehnologie Java 2 Enterprise Edition. Cu această nouă versiune dezvoltatorii de software vor putea opta pentru unul dintre sistemele de operare Windows, Linux sau Solaris.

Noii clienți ai firmei Inprise vor putea achiziționa Borland JBuilder 3.5 Enterprise la un preț de 2499$ si Borland C++Builder 5 Professional la 799$. Actualii posesori ai unui produs Inprise din categoria client/server sau enterprise vor putea achiziționa Borland JBuilder 3.5 Enterprise cu 1699$. În cazul în care aceștia dețin o licență pentru Borland JBuilder 3 prețul se reduce la 799$. Posesorii unui produs Inprise din categoria professional vor putea beneficia de facilitățile Borland JBuilder 5 Enterprise la un preț de 2199$.

Introducere în teoria GRAFURILOR

Cristian Cadar

Parcurgerea DF, prezentată într-un episod anterior al acestui serial are numeroase aplicații. Vă prezentăm acum două dintre ele: determinarea conexității unui graf și sortarea topologică.

Printr-un *graf conex* înțelegem un graf în care între oricare două noduri ale sale există cel puțin un drum. Determinarea conexității unui graf este o problemă deosebit de importantă, pentru a cărei rezolvare există mai mulți algoritmi. O primă variantă ar fi de exemplu crearea matricei drumurilor, așa cum s-a arătat în articolul despre determinarea drumurilor minime (algoritmul *Roy-Warshall*). Având construită această matrice, știm că graful este conex dacă și numai dacă toate elementele matricei sunt egale cu 1 (deci există drum între oricare două noduri ale grafului). De remarcat că această metodă se poate aplica atât grafurilor neorientate cât și celor orientate. Algoritmul în pseudocod este următorul:

```
boolean Conex(graf G)
D ← A    // A este matricea de adiacență
pentru k=1,n
  pentru i=1,n
    pentru j=1,n
      dacă D(i,k)=1 și D(k,j)=1
        D(i,j) ← 1
c ← adevărat
pentru i=1,n
  pentru j=1,n
    dacă D(i,j)=0
      c ← fals
Conex ← c
```

Complexitatea acestui algoritm este $O(n^3)$, complexitate total ineficientă, dată de algoritmul *Roy-Warshall*.

În ceea ce privește grafurile neorientate, determinarea conexității unui graf se poate face foarte simplu și foarte eficient printr-o singură parcurgere *DF* sau *BF* (de obicei se alege *DF*). Parcurgerea se face dintr-un nod oarecare al grafului, iar graful va fi conex dacă și numai dacă toate nodurile acestuia au fost vizitate. Dacă acest lucru se întâmplă, vom avea un drum de la orice nod la rădăcina arborelui *DF*, și prin urmare, vom avea drum între oricare două noduri ale grafului, trecând prin rădăcină. Algoritmul în pseudocod este:

```
DF(nod k)
Vizitat(k) ← adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu Vizitat(i)
    DF(i)
  i ← următorul_vecin(k)
boolean Conex(graf G)
pentru i=1,n
  Vizitat(i) ← fals
DF(1)
c ← adevărat
pentru i=1,n
  dacă nu Vizitat(i)
    c ← fals
Conex ← c
```

Algoritmul are bineînțeles complexitatea $O(n+m)$, complexitate optimă.

În cazul grafurilor neconexe, ne poate interesa aflarea componentelor sale conexe. Printr-o ***componentă conexă*** a unui graf, înțelegem o mulțime maximală de noduri ale grafului, cu proprietatea că între oricare două noduri ale sale există cel puțin un drum. Pentru a afla componentele conexe ale unui graf neorientat se procedează în felul următor: mai întâi se face o parcurgere *DF* din nodul 1, determinându-se componenta conexă în care se află acest nod, apoi se face o nouă parcurgere dintr-un nod nevizitat încă, determinându-se componenta conexă din care acesta face parte, și tot așa până când au fost vizitate toate nodurile:

```
DF(nod k)
afișează k
Vizitat(k) ← adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu Vizitat(i)
    DF(i)
  i ← următorul_vecin(k)
Componente_Conexe(graf G)
pentru i=1,n
  Vizitat(i) ← fals
pentru i=1,n
  dacă nu Vizitat(i)
    afișează "O nouă componentă conexă:"
    DF(i)
```

În ceea ce privește grafurile orientate, determinarea conexității (care în acest caz se numește ***tare conexitate***) este ceva mai dificilă. Așa cum am amintit mai sus, algoritmul *Roy-Warshall* rămâne valabil și în acest caz, însă complexitatea lui ($O(n^3)$) este mult prea mare.

Un algoritm mai eficient de determinare a componentelor tare conexe ale unui graf orientat (și implicit de determinare a tare conexității grafului) este algoritmul *plus-minus*, care acționează în felul următor: se face o parcurgere *DF* dintr-un nod oarecare al grafului (alegem de obicei nodul 1). Se marchează cu *plus* toate nodurile la care s-a ajuns în această parcurgere. Apoi, se face o nouă parcurgere *DF*, plecând din același nod, avansându-se acum pe muchiile care intră în noduri. De această dată, toate nodurile la care s-a ajuns în această parcurgere se marchează cu *minus*. De fapt, a doua parcurgere se face pe graful transpus al lui *G*. Acest graf se notează G^T și se obține din graful *G* prin inversarea sensurilor muchiilor. De exemplu, figura 2 conține graful transpus al grafului din figura 1.

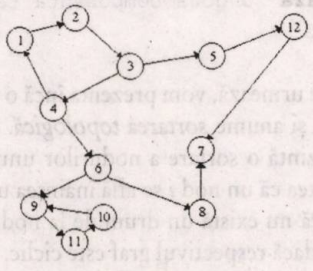


Figura 1: Un graf

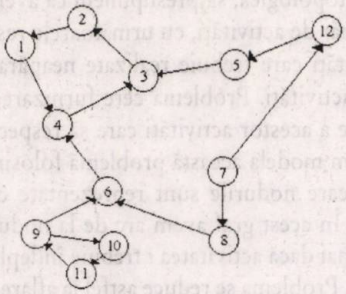


Figura 2: Graful transpus

Revenind la algoritmul *plus-minus*, după ce realizăm cele două parcurgeri, putem afla componenta tare conexă din care face parte nodul 1; aceasta este alcătuită din nodurile care au fost marcate atât cu *plus* cât și cu *minus*. Explicația este simplă: cu *plus* sunt marcate nodurile la care se poate ajunge din nodul 1, iar cu *minus* sunt marcate nodurile din care se poate ajunge în nodul 1. Bineînțeles, componenta tare conexă a nodului 1 este reprezentată de intersecția celor două mulțimi, deoarece astfel vom avea drum între oricare două noduri, în ambele sensuri, trecând prin nodul 1.

În figura 3, se poate remarca modul în care au fost etichetate nodurile grafului din figura 1, atunci când parcurgerea se face din nodul 1:

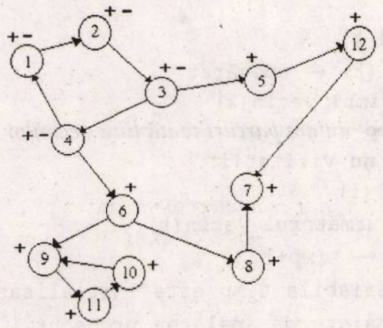


Figura 3: Algoritmul plus-minus

Pentru a afla și celelalte componente tare conexe ale grafului, repetăm pașii de mai sus pentru nodurile care nu au fost incluse încă în nici o componentă tare conexă. Iată descrierea în pseudocod a algoritmului *plus-minus*:

```
DF_Plus_Minus(nod k, graf G, marcaj)
marchează cu 'marcaj' nodul k
i ← primul_vecin(k)
cât timp nu am parcurs toata lista vecinilor lui k
  dacă nu Vizitat(i)
    DF_Plus_Minus(i,G,marcaj)
  i ← următorul_vecin(k)
Plus_Minus(graf G)
pentru i=1,n
  v(i) ← fals
pentru i=1,n
  dacă nu v(i)
    afișează "O nouă componentă tare conexă:"
    Vizitat ← fals
    DF_Plus_Minus(i,G,+)
    Vizitat ← fals
    DF_Plus_Minus(i,G^T,-)
    pentru i=1,n
    dacă i este marcat cu plus și cu minus
      v(i) ← adevărat
      afișează i
```

Acest algoritm are, în cazul mediu, complexitatea $O(n+m)$, însă în cazul cel mai defavorabil atinge o complexitate de $O(n^2)$. În figura 4 este ilustrat un astfel de caz:

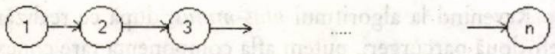


Figura 4: Algoritmul plus-minus are complexitatea $O(n^2)$

Algoritmul optim de aflare a componentelor tare conexe ale unui graf are complexitatea $O(n+m)$ și se bazează pe algoritmul *plus-minus*. Acest algoritm realizează, de asemenea, două parcurgeri *DF*, una pe graful G inițial, iar cealaltă pe graful transpus G^T. Diferența față de algoritmul anterior este că cea de-a doua parcurgere se face în funcție de timpii finali din prima parcurgere, asigurându-se astfel că fiecare nod este parcurs înaintea succesorilor săi. Acești timpi reprezintă timpii la care am terminat prelucrarea fiecărui nod. Determinarea lor se face în felul următor:

```
DF1(nod k)
Vizitat(k) ← adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu Vizitat(i)
    DF1(i)
  i ← următorul_vecin(k)
  timp ← timp+1
  // Variabila timp este inițializată cu 0
  // înainte de apelarea procedurii
Final(k) ← timp
```

Având timpii finali memorați în vectorul `Final`, parcurgerile *DF* pe graful transpus se vor face începând cu vârful cu cel mai mare timp final și continuând în ordine descrescătoare cu celelalte noduri care nu au fost incluse încă într-o componentă tare conexă. La fiecare dintre aceste parcurgeri, toate nodurile marcate formează o componentă tare conexă.

Spre exemplificare, să considerăm din nou graful din figura 1. Mai întâi, printr-o parcurgere *DF*, aflăm timpii finali pentru fiecare nod. Aceștia sunt:

Nod	Final
1	12
2	11
3	10
4	7
5	9
6	6
7	1
8	2
9	5
10	3
11	4
12	8

Parcurgerea pe graful transpus se va face în ordine descrescătoare a timpilor finali. Se începe cu nodul 1; nodurile la care se ajunge din nodul 1 în graful G^T sunt 1, 4, 3 și 2 care formează o primă componentă tare conexă. Din vârfurile rămase, îl alegem pe cel cu cel mai mare timp final, și anume vârful 5. Din acest vârf nu putem ajunge în nici un alt vârf neinclus încă într-o componentă tare conexă, deci el formează singur o componentă tare conexă. Similar pentru nodul 6. Apoi începem o nouă parcurgere din nodul 9 și aflăm componenta tare conexă formată din nodurile 9, 11și 10. Apoi pornim din nodul 8, aflând componenta tare conexă formată numai din acest nod și în final din nodul 7, aflând componenta tare conexă formată de asemenea numai din acest nod. Algoritmul descris în pseudocod va fi:

```
DF(nod k, graf G)
afișează_nodul_k
v(k) ← adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu V(i)
    atunci DF(i,G)
  i ← următorul_vecin(k)

Tare_conexe(graf G)
pentru i=1,n
  Vizitat(i) ← fals
timp ← 0
pentru i=1,n
  dacă nu Vizitat(i)
    DF(i,G)
pentru i=1,n
  v(i) ← fals
  // v(i) indică dacă nodul i a fost inclus
  // într-o componentă tare conexă
pentru fiecare nod k din graf
      (parcurse în ordine descrescătoare a timpilor finali)
  dacă nu v(i)
    afișează "O nouă componentă tare conexă:"
    DF(i,G^T)
```

În cele ce urmează, vom prezenta încă o aplicație a parcurgerii *DF*, și anume ***sortarea topologică***. Sortarea topologică reprezintă o sortare a nodurilor unui graf orientat cu proprietatea că un nod i se află înaintea unui nod j dacă și numai dacă nu există un drum de la nodul j la nodul i. Bineînțeles, dacă respectivul graf este ciclic, acesta nu poate fi sortat topologic. Pentru a înțelege mai bine conceptul de sortare topologică, să presupunem că avem de îndeplinit un număr de activități, cu următoarele restricții: pot exista activități care trebuie realizate neapărat înainte sau după alte activități. Problema cere furnizarea unei ordini de realizare a acestor activități care să respecte restricțiile cerute. Vom modela această problemă folosind un graf orientat în care nodurile sunt reprezentate de activitățile respective. În acest graf avem arc de la nodul i la nodul j dacă și numai dacă activitatea i trebuie îndeplinită înaintea activității j. Problema se reduce astfel la aflarea unei sortări topologice a acestui graf.

În primul rând trebuie să determinăm dacă graful este sau nu aciclic. Acest fapt presupune ca în parcurgerea *DF* să nu existe muchii de întoarcere, adică muchii care să ajungă într-un nod deja vizitat.

Dacă graful este aciclic, atunci există sigur o sortare topologică, aceasta fiind dată chiar de sortarea nodurilor în ordine descrescătoare a timpilor finali. Pentru a păstra complexitatea $O(n+m)$, vectorul de noduri sortate topologic va fi realizat prin adăugarea fiecărui nod la începutul unei cozi. Algoritmul în pseudocod este următorul:

```
DF(nod k)
afișează_nodul_k
Vizitat(k) ← adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu Vizitat(i)
    DF(i)
    altfel afișează "Graful este ciclic"
  i ← următorul_vecin(k)
  c ← i       // introduc nodul i în coada c

Topologic (graf G)
pentru i=1,n
  Vizitat(i) ← fals
timp ← 0
pentru i=1,n
  dacă nu Vizitat(i)
    DF(i)
afișează_coada_C
```

Corectitudinea acestui algoritm este ușor de intuit: într-un graf orientat aciclic un nod *i* are un timp final mai mare decât timpul final al unui alt nod *j*, dacă și numai dacă prelucrarea nodului *j* a fost terminată înainte de a fi terminată prelucrarea nodului *i*, adică de la nodul *j* nu există arce spre nodul *i*.

De remarcat că ordinea în care sunt parcurse nodurile pentru determinarea tare conexității (algoritmul $O(n+m)$) este chiar ordinea furnizată de sortarea topologică.

O altă metodă de realizare a sortării topologice este următoarea:

- afișăm nodurile de grad interior 0 (deci nodurile în care nu intră nici un arc);
- eliminăm aceste noduri și arcurile incidente din graf;
- reactualizăm gradele nodurilor grafurilor;
- repetăm pașii anteriori fie până când am afișat toate nodurile, fie până când nu mai există noduri de grad interior 0 (graful conține cicluri și nu poate fi sortat topologic).

O primă variantă de implementare ar fi următoarea:

```
Topologic(graf G)
pentru k=1,n
  d(k) ← 0
  // vectorul d reține gradele vârfurilor
pentru k=1,m
  d(i) ← d(i)+1
  // muchia a k-a este muchia (i,j)
  d(j) ← d(j)+1
nr ← 0
cât timp nr<n    // nu am afișat toate nodurile
  găsit ← fals
  pentru i=1,n
    dacă d(i)=0
      găsit ← adevărat
      afișează i
      nr ← nr+1
      j ← primul_vecin(i)
      cât timp nu am parcurs toată
                          lista vecinilor lui i
        d(j) ← d(j)-1
                    // reactualizez gradele
        j ← următorul_vecin(i)
                      // nodurilor grafului
  dacă nu găsit
   afișează "Eroare! - Graful este ciclic !!!"
```

Se remarcă ușor că algoritmul de mai sus are complexitatea $O(n^2)$, complexitate dată de căutarea nodurilor cu gradul interior 0. Pentru a-l aduce la complexitatea $O(n+m)$, este suficient să folosim o structură de coadă în care să reținem nodurile cu gradul interior 0. Algoritmul în pseudocod devine:

```
Topologic(graf G)
pentru k=1,n
  d(k) ← 0
  // vectorul d reține gradele vârfurilor
pentru k=1,m
  d(i) ← d(i)+1
  // muchia a k-a este muchia (i,j)
  d(j) ← d(j)+1
pentru i=1,n
  dacă d(i)=0
  c ← i
  // se introduc în coadă nodurile cu gradul
  // interior 0
nr ← 0
cât timp c nu este vidă și nr<n
  k ← primul nod din c
  afișează k
  nr ← nr+1
  i ← primul_vecin(k)
  cât timp nu am parcurs toată lista vecinilor lui k
    d(i) ← d(i)-1   // se reactualizează
  dacă d(i)=0       // gradele nodurilor
    c ← i           // grafului și coada
  următorul_vecin(k)
dacă nr<n
  afișează "Eroare! - Graful este ciclic !!!"
```

Cristian Cadar este student în anul II la Universitatea Politehnică din București. Poate fi contactat la adresa ccadar@pcnet.pcnet.ro.

Pagini WEB cu XML 1.0

Mihai Scorțaru

Chiar dacă, de obicei, limbajul XML nu este folosit direct pentru crearea paginilor, proiectanții paginilor Web trebuie să cunoască acest standard pentru a crea site-uri profesionale. XML pare, în acest moment, singura noastră șansă de a obține un format pentru transferul informațiilor care să fie, cu adevărat, independent de platformă. În următoarele câteva episoade ale acestui serial vă vom prezenta conceptele care stau la baza acestui limbaj.

XML este prescurtarea de la *eXtensible Markup Language* și este, în principal, un format pentru stocarea datelor. La ora actuală XML a fost adoptat de majoritatea companiilor din industria IT.

Singurul moment în care s-a mai obținut o astfel de uniformitate a fost atunci când TCP/IP a fost adoptat ca protocol pentru rețele. XML este un standard internațional și se bucură de susținere din partea companiilor din întreaga lume.

S-ar putea să pară bizar faptul că, într-o lume în care standardele apar și dispar precum anotimpurile, în care chiar și standardele sunt fragmentate, extinse sau particularizate de către companii în scopuri comerciale, totuși XML nu respectă această regulă. Surprinzător, toate aceste companii încearcă să obțină și să se supună unui singur standard.

Diferențe față de HTML

XML, la fel ca și HTML, folosește marcaje; diferența esențială este aceea că, în XML, marcajele sunt folosite pentru a descrie structura unui text și nu modul în care acest text va fi afișat. XML nu are o colecție fixă de marcaje; să considerăm următorul exemplu:

```
<BODY>
  3DO anunta:
  <H3>
    Might and Magic VIII
  </H3>
  Noua versiune a fost lansata pe data de
  <B> 29 februarie 2000 </B>.
</BODY>
```

Salvând textul anterior într-un fișier cu extensia `.html` (de exemplu, **MM8.HTML**) și vizualizând fișierul prin intermediul unui program de navigare vom vedea o pagină Web asemănătoare cu cea din figura 1.

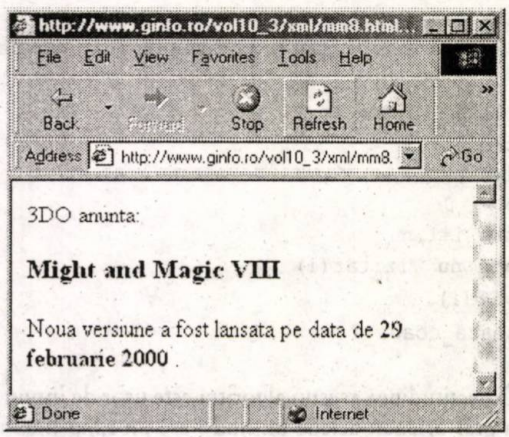


Fig.1 MM8.HTML în Internet Explorer 5.0

Așa cum v-ați așteptat, programul de navigare va interpreta documentul HTML și va genera pagina Web corespunzătoare. Totuși, dacă vom salva documentul cu extensia `.xml` (de exemplu, **MM8.XML**) rezultatul obținut este asemănător cu cel din figura 2.

Fig. 2 MM8.XML în Internet Explorer 5.0

Fișierul **MM8.XML** este interpretat astfel:

- Avem un marcaj **<BODY>** care conține un text
- În acest text avem un marcaj **<H1>** care conține text și un marcaj **<B>** care conține tot text.

Internet Explorer 5.0 nu face decât să interpreteze în acest mod documentul **MM8.XML**, iar această interpretare nu implică și efectuarea unei formatări.

Totuși, programul de navigare realizează un fel de formatare pentru a ușura citirea informațiilor conținute în acest document. El indentează marcajele de pe nivelurile inferioare pentru a asigura o lizibilitate sporită.

Deocamdată, *Internet Explorer 5.0* este singurul program de navigare care oferă suport rezonabil pentru XML.

Denumirea marcajelor

Să considerăm acum exemplul unei agende telefonice. Pentru a descrie conținutul agendei vom putea scrie un document XML asemănător cu următorul:

```
<Agenda>
   <Persoana>
      <Nume>x</Nume>
      <Prenume>y</Prenume>
      <Telefon>z</Telefon>
   </Persoana>
   <Persoana>
      <Nume>xx</Nume>
      <Prenume>yy</Prenume>
      <Telefon>zz</Telefon>
   </Persoana>
<Persoana>
      <Nume>xxx</Nume>
      <Prenume>yyy</Prenume>
      <Telefon>zzz</Telefon>
      </Persoana>
</Agenda>
```

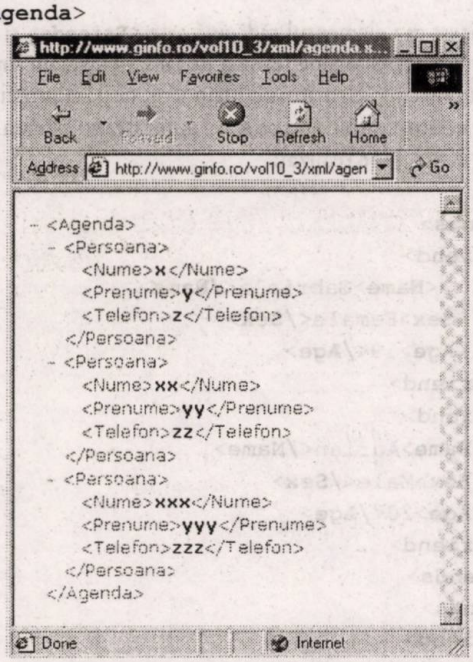


Fig. 3 AGENDA.XML în Internet Explorer 5.0

Avem aici o colecție de marcaje și ați putea crede că denumirile acestora au o anumită semnificație deoarece sunt intuitive. Este destul de evident că documentul anterior conține o agendă care conține o serie de persoane și pentru fiecare persoană avem numele, prenumele și numărul de telefon. "Problema" limbajului XML este aceea că aceste marcaje nu au nici o semnificație pentru el. După cum se poate vedea în figura 3, *Internet Explorer 5.0* nu interpretează în nici un fel aceste marcaje, ci doar le afișează.

De fapt nu există nici un motiv pentru care să nu descriem conținutul agendei astfel:

```
<Ciudat>
   <Abracadabra>
      <Hocus>x</Hocus>
      <Pocus>y</Pocus>
      <Preparatus>z</Preparatus>
   </Abracadabra>
   <Abracadabra>
      <Hocus>xx</Hocus>
      <Pocus>yy</Pocus>
      <Preparatus>zz</Preparatus>
   </Abracadabra>
   <Abracadabra>
      <Hocus>xxx</Hocus>
      <Pocus>yyy</Pocus>
      <Preparatus>zzz</Preparatus>
   </Abracadabra>
</Ciudat>
```

Programul de navigare va afișa, și de această dată, numai marcajele și textul, fără să le interpreteze (figura 4).

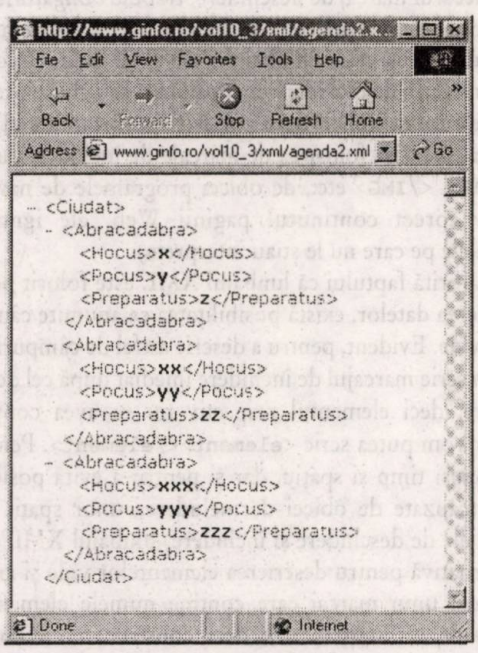


Fig. 4 AGENDA2.XML în Internet Explorer 5.0

În concluzie, nu contează cum denumim marcajele, semnificația acestora fiind determinată numai de modul în care le folosim; bineînțeles, este recomandabil ca marcajele să aibă denumiri sugestive.

Datorită faptului că fișierele .XML sunt fișiere text standard, ele pot fi foarte ușor transferate de la un calculator la altul, deci poate fi folosit ca format universal pentru schimbul de informații. Mai mult, nu este un format care să fie proprietate a unei companii, deci poate fi folosit de oricine, și dacă marcajele au denumiri sugestive, datele păstrate în format XML se autodescriu.

Marcaje și elemente

În cazul limbajului HTML prin *marcaj* înțelegem, de obicei, un text care transmite programului de navigare anumite instrucțiuni pentru formatarea conținutului unei pagini Web (de exemplu, <B>, <OBJECT>, <FORM>). Prin *element* înțelegem întregul text care descrie formatarea unei componente a unei pagini Web. De obicei, acest text cuprinde marcajul de deschidere, textul care descrie componenta și marcajul de închidere. De exemplu, textul <B>The Romanian Independency was proclaimed on <I>May 10th 1877</I></B> conține 6 marcaje, dar numai trei elemente. Textul corespunzător elementului SUP este th, cel corespunzător elementului I este <I>May 10th 1877</I>, iar cel corespunzător elementului B este <B>The Romanian Independency was proclaimed on <I>May 10th 1877</I></B>.

Deci, un element constă dintr-un marcaj de deschidere, unul de închidere și textul cuprins între acestea, text care poate conține și alte elemente. Aceasta este o caracteristică fundamentală a limbajului XML și introduce conceptul de *document XML bine realizat*, document în care fiecărui marcaj de deschidere, *trebuie obligatoriu* să îi corespundă un marcaj de închidere. Acest lucru nu era obligatoriu în cazul limbajului HTML unde folosirea unor marcaje de închidere era opțională sau chiar interzisă în unele cazuri (chiar dacă potrivit specificațiilor HTML 4.0 era interzisă folosirea de marcaje de închidere cum ar fi </BR>, </IMG> etc., de obicei programele de navigare afișau corect conținutul paginii Web, ele ignorând marcajele pe care nu le știau interpreta).

Datorită faptului că limbajul XML este folosit pentru descrierea datelor, există posibilitatea ca anumite câmpuri să fie vide. Evident, pentru a descrie astfel de câmpuri vide putem scrie marcajul de închidere imediat după cel de deschidere, deci elementul respectiv nu va avea conținut; așadar vom putea scrie <element></element>. Pentru a economisi timp și spațiu, dar și pentru a evita posibilele erori (cauzate de obicei de includerea unor spații între marcajele de deschidere și închidere) limbajul XML oferă o alternativă pentru descrierea elementelor vide și anume folosirea unui marcaj care conține numele elementului urmat de un *slash* ('/'); deci construcția <element></element> este echivalentă cu <element/>.

Spre deosebire de limbajul HTML, XML face distincție între literele mici și literele mari care fac parte din denumirea marcajelor. Pentru ca un document XML să fie bine realizat ***nu este permis*** ca în marcajul de închidere al unui element să înlocuim o literă mică cu majuscula corespunzătoare sau invers.

Elemente rădăcină

Elementul rădăcină este marcajul care cuprinde în interiorul său toate celelalte elemente prezente în document. Este evident faptul că un document XML bine realizat poate conține *o singură* rădăcină. Să considerăm exemplul:

```
<Friends>
  <Friend>
      <Name>Gabriela</Name>
    <Sex>Female</Sex>
    <Age>19</Age>
  </Friend>
  <Friend>
    <Name>Adrian</Name>
    <Sex>Male</Sex>
    <Age>20</Age>
  </Friend>
  <Friend>
    <Name>Carmen</Name>
    <Sex>Female</Sex>
    <Age>36</Age>
  </Friend>
  <Friend>
    <Name>Gelu</Name>
    <Sex>Male</Sex>
    <Age>20</Age>
  </Friend>
</Friends>
```

În acest caz elementul rădăcină este <Friends> și documentul XML este bine realizat deoarece există un singur element rădăcină. Următorul document, deși, în principiu, conține aceleași informații ca și cel anterior, nu este un document XML bine realizat:

```
<Friends>
  <Friend>
      <Name>Gabriela</Name>
    <Sex>Female</Sex>
    <Age>19</Age>
  </Friend>
  <Friend>
    <Name>Adrian</Name>
    <Sex>Male</Sex>
    <Age>20</Age>
  </Friend>
</Friends>
<Friends>
  <Friend>
    <Name>Carmen</Name>
```

```
    <Sex>Female</Sex>
    <Age>36</Age>
  </Friend>
  <Friend>
    <Name>Gelu</Name>
    <Sex>Male</Sex>
    <Age>20</Age>
  </Friend>
</Friends>
```

Pe primul nivel există două elemente, deci documentul XML nu este bine realizat chiar dacă aceste două elemente au aceeași denumire.

Marcajul <?xml>

Acest marcaj nu este un marcaj XML veritabil, ci unul special care este folosit pentru a indica instrucțiuni de procesare. Acest marcaj trebuie să apară pe prima linie a oricărui document XML și este folosit pentru a indica versiunea XML folosită precum și informații despre setul de caractere.

Versiunea poate fi indicată prin intermediul atributului **version** al marcajului <?xml>. Prin `<?xml version="1.0">` indicăm faptul că versiunea XML folosită este 1.0. În acest moment XML 1.0 este versiunea implicită și singura disponibilă. Totuși, este destul de probabil ca, în viitor, să apară alte versiuni ale limbajului XML.

Cu ajutorul atributului encoding putem specifica setul de caractere folosit pentru descrierea datelor. Specificarea setului de caractere este necesară în cazul în care sunt folosite caractere care nu fac parte din setul de caractere ASCII. Cele mai frecvente seturi de caractere folosite sunt:

- `UTF-8` - *Unicode*
- `ISO-8859-1` - Latin 1 (Western Europe, Latin America)
- `ISO-8859-2` - Latin 2 (Central/Eastern Europe)
- `ISO-8859-3` - Latin 3 (SE Europe)
- `ISO-8859-4` - Latin 4 (Scandinavia/Baltic)
- `ISO-8859-5` - Latin/Cyrilic
- `ISO-8859-6` - Latin/Arabic
- `ISO-8859-7` - Latin/Greek
- `ISO-8859-8` - Latin/Hebrew
- `ISO-8859-9` - Latin/Turkish
- `ISO-8859-10` - Latin/Lappish/Nordic/Eskimo
- EUC-JP sau `Shift-JIS` - Japanese

Deci, pentru a putea folosi caractere românești în descrierea unor date, prima linie a documentului XML pe care îl creați va trebui să fie `<?xml version="1.0" encoding="iso-8859-2">`.

Atribute

La fel ca și în cazul limbajului HTML, marcajele XML pot avea atribute pentru a defini proprietățile elementelor. Pentru ca un document HTML să fie bine realizat *este obligatoriu* ca valorile atributelor să fie incluse între ghilimele (`'"'`). Ați putut vedea modul în care se pot folosi atributele în cazul marcajului <?xml>; pentru a caracteriza mai bine datele cuprinse într-un document XML și alte elemente pot avea atribute. De exemplu, pentru a descrie o carte putem avea:

```
<Carte ISBN="xxx-xxxxx-x-x">
  <Autor>xxx</Autor>
  <Titlu>xxx</Titlu>
  <Pagini>xxx</Pagini>
</Carte>
```

Limbajul HTML permite indicarea valorilor atributelor fără a le mai include între ghilimele, dar acest lucru nu este permis într-un document XML bine realizat.

Caractere speciale

Există câteva caractere speciale care nu pot să apară în interiorul unor șiruri de caractere care descriu date. Aceste caractere trebuie înlocuite prin secvențe de caractere cu semnificație echivalentă. Caracterele speciale sunt:

- `&` - înlocuit cu `&`
- `<` - înlocuit cu `<`
- `>` - înlocuit cu `>`
- `"` - înlocuit cu `"`
- `'` - înlocuit cu `'`

De exemplu, construcția

```
<Titlu> Bac Info' 2000 </Titlu>
```

nu este permisă.

Pentru a avea un document XML bine realizat va trebui să scriem

```
<Titlu> Bac Info&apos; 2000 </Titlu>.
```

Scheme și DTD-uri

Am afirmat anterior că marcajele XML nu au nici o semnificație și ele pot avea orice denumire. Totuși, trebuie să știm care sunt marcajele care pot apărea în cadrul unui document XML. Pentru a realiza acest lucru va trebui să folosim fie o *schemă*, fie o *definiție de tipuri pentru document* (engl. *Document Type Definition* - DTD). Cu ajutorul schemelor sau al definițiilor de tipuri un document XML bine realizat poate fi transformat într-un *document XML valid*. Un document XML valid, pe lângă faptul că este bine realizat, conține numai elementele și atributele permise.

La ora actuală există atât scheme cât și DTD-uri deoarece, la un moment dat, firma *Microsoft* a considerat că ideea care stă la baza DTD-urilor este greșită, motiv pentru care a introdus schemele.

Un DTD este un fișier text care definește structura unui document XML fără a folosi acest limbaj. În cazul schemelor, structura documentului este definită folosind chiar limbajul XML.

Să considerăm un DTD tipic care definește structura unui document XML care conține date referitoare la autorii unei cărți.

```
<!ELEMENT DOCUMENT (AUTOR+)>
<!ELEMENT AUTOR(cod, nume, prenume, tel, str,
                nr, ap,oras,judet)>
<!ELEMENT cod(CDATA)>
<!ELEMENT nume(CDATA)>
<!ELEMENT prenume(CDATA)>
<!ELEMENT tel(CDATA)>
<!ELEMENT str(CDATA)>
<!ELEMENT nr(CDATA)>
<!ELEMENT ap(CDATA)>
<!ELEMENT oras(CDATA)>
<!ELEMENT judet(CDATA)>
```

Structura acestui fișier este destul de simplă. Este indicat faptul că documentul conține 0 sau mai multe elemente AUTOR și fiecare element AUTOR conține nouă alte elemente, fiecare dintre aceste subelemente conținând caractere (CDATA).

Există două dezavantaje în folosirea DTD-urilor: nu sunt scrise în XML și nu permit specificarea tipurilor datelor (întregi, date calendaristice etc.). Din aceste motive *Microsoft* a propus consorțiului *World Wide Web (W3C)* introducerea schemelor. Convertind DTD-ul anterior într-o schemă, vom obține:

```
<Schema ID="Autor">
  <Element name="cod" type="string"/>
  <Element name="nume" type="string"/>
  <Element name="prenume" type="string"/>
  <Element name="telefon" type="string"/>
  <Element name="str" type="string"/>
  <Element name="nr" type="string"/>
  <Element name="ap" type="string"/>
  <Element name="oras" type="string"/>
  <Element name="judet" type="string"/>
</Schema>
```

Unicul tip permis în DTD-uri (CDATA) este echivalent cu tipul string în scheme, dar schemele permit folosirea și a altor tipuri de date. De exemplu, pentru a indica un element care poate avea numai valori logice vom scrie <Element name="*denumire*" type="boolean"/>. Este important să precizăm faptul că schemele nu au fost încă standardizate, motiv pentru care nu vom insista momentan asupra lor.

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro.

news...

VIA Cyrix III

Compania VIA Technologies a anunțat lansarea procesorului VIA Cyrix III, primul procesor Socket 370 din lume care furnizează caracteristici și performanțe "premium" pentru segmentul de piață Value PC. Lansat la viteze de până la PR533 (echivalent cu un Pentium la 533 MHz), procesorul VIA Cyrix III este singura soluție ce livrează caracteristici de performanță îmbunătățite pentru segmentul Value PC, inclusiv Front Side Bus de 133 MHz, cache de 256K L2 și tehnologie 3DNowI. Compatibil în totalitate cu seria plăcilor de bază Socket 370, procesorul VIA Cyrix III furnizează o soluție competitivă din punctul de vedere al raportului preț/performanță.

Photopaint 9

După ce a lansat propria sa versiune a sistemului de operare Linux, compania Corel continuă să contribuie la impunerea pe piață a acestuia.

Corel Corporation a anunțat că va oferi gratuit pe Internet produsul Corel Photo-Paint 9 pentru Linux. Produsul va putea fi descărcat, la începutul verii și va oferi posibilități de editare a fotografiilor digitale, de creare de imagini și de realizare de picturi.

Versiunea pentru Linux a produsului Corel Photopaint 9 oferă suport pentru majoritatea formatelor de imagini răspândite la ora actuală, precum și o soluție simplă pentru publicare pe Web.

Microsoft pe locul II

În data de 24 martie 2000, pentru prima dată în istoria firmei, Cisco Systems a devenit cea mai valorasă companie din lume. Valoarea totală a acțiunilor Cisco Systems a atins 575,93 miliarde de dolari, cea a acțiunilor Microsoft fiind de 575,29 miliarde de dolari. Acest lucru dovedește superioritatea conceptelor și politicilor de firmă bazate pe Internet, față de abordarea tradițională bazată pe evoluția PC-ului și a sistemelor de operare PC.

REȚELE de calculatoare și mediul INTERNET

Marin Vlada

În ultimii ani Internet-ul are o influență tot mai mare asupra societății umane. Vă vom prezenta în continuare câteva dintre tehnologiile și standardele care au făcut posibilă apariția unei astfel de rețele. De asemenea veți putea afla și cum funcționează această rețea.

Rețele de calculatoare

Apariția calculatoarelor conectate în rețea a reprezentat o performanță privind utilizarea calculatoarelor în procesarea informațiilor. Multiplele avantaje oferite de rețelele de calculatoare sunt:

- utilizare eficientă prin partajarea resurselor unității centrale (UC);
- acces direct la resursele hardware (memorii externe, imprimante etc.) și software (editoare, limbaje de programare, programe specializate);
- păstrarea programelor și fișierelor într-o singură copie (pe server) și utilizarea lor de către oricine cu drept de acces;
- sistem de protecție a fișierelor și programelor;
- utilizarea simultană a bazelor de date de către mai mulți utilizatori;
- comunicare și schimb de informații (programe și fișiere) între utilizatori la nivel local, regional sau mondial.

Dezvoltarea rețelelor de calculatoare (mici sau mari) a contribuit la elaborarea și utilizarea unor aplicații cu un impact mare asupra vieții economice și sociale din întreaga societate:

- accesul la programe complexe ce oferă informații utile la nivel macroeconomic;
- accesul la mari baze de date din domeniile economic, financiar, comunicații etc.;
- accesul la informații științifice și transferul rapid al articolelor științifice.

Astăzi, vechiul model al unui singur calculator (sistem de calcul) care rezolvă problemele (cererile) unui număr de utilizatori, este înlocuit cu modelul nou al unui sistem de calculatoare interconectate. Unul sau mai multe calculatoare dintr-o rețea reprezintă calculatoare centrale (FS - *File Server*) care controlează și gestionează prin intermediul unui sistem de operare întreaga activitate de calcul din rețea. Aceste calculatoare centrale formează nucleul rețelei.

Elemente...

Rețea de calculatoare

- un ansamblu de calculatoare (sisteme de calcul) interconectate prin intermediul unor medii de comunicație (cablu coaxial, fibră optică, linie telefonică, ghid de unde) în scopul utilizării în comun de către un număr foarte mare de utilizatori - chiar la nivel mondial - a tuturor resurselor fizice (hardware), logice (software de bază și aplicații) și informaționale (baze de date) asociate calculatoarelor din rețea. Prin astfel de rețele de calculatoare se poate asigura o integrare informatică a unui număr foarte mare de utilizatori la nivel local, regional și mondial.

Mediul Internet

- un ansamblu de rețele de calculatoare interconectate care împreună cu miile de calculatoare individuale și milioanele de utilizatori, constituie comunitatea *Internet* care are la dispoziție un mediu informațional și de calcul cu foarte multe servicii informatice și resurse; constituie un ansamblu complex de arhitecturi, componente fizice și sisteme de operare, care tratează în mod egal atât minicalculatoarele sau rețelele de calculatoare, cât și PC-urile sau mainframe-urile. Facilitățile mediului au o importanță egală, iar componentele sale se comportă una față de alta fără nici o discriminare.

Calculatorul central poate fi un calculator obișnuit pe care este instalat un sistem de operare pentru rețea: *Netware*, *Unix*, *Linux*, *OS/2*, *Windows NT/2000*. Calculatorul central controlează toate resursele comune (unități de discuri, unități de dischete, imprimante, plottere, modemuri, fișiere etc.), asigură securitatea datelor și sistemului, realizează comunicații între stațiile de lucru.

Din punct de vedere hardware, o rețea de calculatoare este constituită din:

- calculatoare centrale (*Servere*);
- calculatoare de lucru (*terminale*, *stații de lucru* - engl. *Workstations*);
- echipamente periferice;
- elemente de conectare.

Stația de lucru (*Workstation*) este un calculator obișnuit care lucrează sub un sistem de operare obișnuit (*Windows*, *Dos*, *Unix*, *Linux* etc.) și care este folosit de utilizatori obișnuiți. Stațiile de lucru (*terminale*) reprezintă componentele de bază ale unei rețele de calculatoare. O stație de lucru are în configurare o placă de rețea (de exemplu NIC - *Netware Interface Card*) care realizează interfața cu rețeaua, fiind conectate cu restul componentelor din rețeaua de calculatoare.

Sistemele de operare în rețea trebuie să recunoască aceste componente de conectare. De aceea, firmele constructoare oferă drivere corespunzătoare pentru sistemele de operare utilizate cel mai frecvent.

Pe lângă sistemul de operare destinat acțiunilor la nivelul stației de lucru, există programe speciale de comunicații în rețea (de exemplu, pentru sistemul de operare *NetWare* există *Netware Shell*) care permit comunicarea stației de lucru cu calculatorul central și cu toate celelalte stații de lucru conectate la rețea. Aceste programe speciale permit ca toate stațiile de lucru din rețea să utilizeze programele și fișierele de date de pe calculatorul central în funcție de prioritățile recunoscute utilizatorului respectiv.

Tipuri de rețele de calculatoare

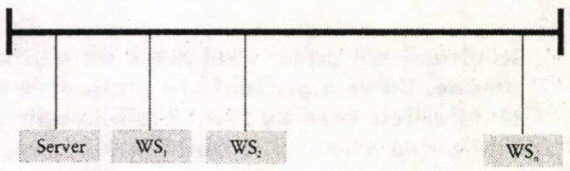
Rețelele de calculatoare sunt de mai multe feluri în funcție de numărul stațiilor de lucru și de aria lor de răspândire:

- rețele locale (LAN - *Local Area Networks*) - lucrează la nivelul unei clădiri sau al unui grup de clădiri având distanța între stațiile de lucru de 10-1000 m;
- rețele metropolitane (MAN - *Metropolitan Area Networks*) - lucrează la nivelul unui oraș având distanța între stațiile de lucru de ordinul kilometrilor;
- rețele regionale/mondiale (WAN - *Wide Area Networks*) - lucrează la nivelul unei regiuni sau la nivel mondial având distanța între stațiile de lucru de ordinul miilor de kilometri;
- rețele publice (PDN - *Public Data Networks*) - lucrează la nivelul unei regiuni sau la nivel mondial și au acces la diverse rețele locale, de exemplu
  - *Internet* (*e-mail* - poștă electronică, WWW - *World Wide Web*);
  - *Usenet* și *Eunet* (poșta electronică și circulația știrilor);
  - *Csnet* și *Arpanet* (cercetare științifică);
  - *Bitnet* (informații în diverse domenii).

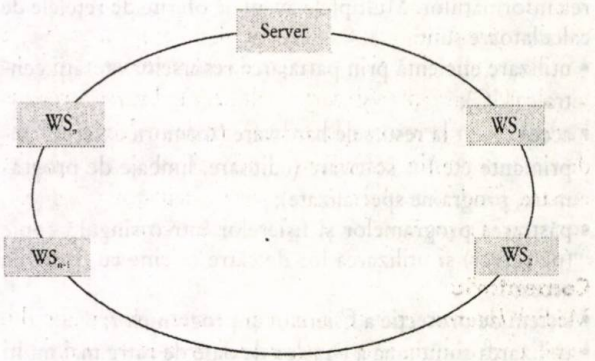
Topologii (structuri) de rețele locale

Topologia (structura) unei rețele de calculatoare reprezintă modul de conectare a stațiilor de lucru față de *Server*. Modul de conectare va determina traseul pe care va circula informația în rețea, prin urmare o structură a comunicațiilor în rețea. În general, există trei tipuri de topologii pentru rețelele LAN:

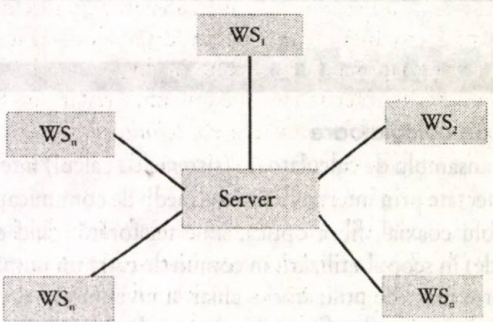
- topologia *Bus* (magistrala) - are o fiabilitate sporită, viteză mare de transmisie;



- topologia *Ring* (inel) - permite ca stațiile de lucru să fie "egale" între ele;



- topologia *Star* (stea) - oferă o viteză mare de comunicație, fiind destinată aplicațiilor în timp real.



În domeniul topologiilor pentru rețele de calculatoare s-au realizat și unele standardizari, dintre care amintim:

- ETHERNET - topologie *Bus* (maxim 150 stații de lucru) produsă de firma *Xerox* în colaborare cu firmele *Intel* și *Digital*;

- RX-Net (maxim 255 stații de lucru), IPX, X25;
- IBM Token Ring - topologie Ring (maxim 96 stații de lucru);
- IBM-PC - topologie Bus;
- ARCNET, MICOM, GATEWAY, G/NET.

Mediul Internet și Rețeaua (The Net)

Spre deosebire de rețelele comerciale de calculatoare pentru servicii (*CompuServe*, *America Online*, *GEnie* etc.), mediul *Internet* este un ansamblu de rețele de calculatoare la nivel mondial care realizează conectarea a milioane de calculatoare și care reprezintă:

- calculatoare guvernamentale și ale instituțiilor națiunilor din întreaga lume;
- calculatoare ale miilor de universități și școli din întreaga lume;
- sisteme de calcul ale marilor corporații (*Microsoft*, *IBM*, *Digital* etc.);
- sisteme de calcul ale unor organizații și fundații științifice, culturale, sociale;
- calculatoare ale marilor agenții de presă și de televiziune din întrega lume;
- calculatoare ale marilor edituri din lume;
- calculatoare ale milioanelor de utilizatori din întreaga lume.

Din punct de vedere arhitectural, mediul *Internet* este de fapt o rețea gigantică de rețele de calculatoare răspândită la nivel mondial și care înglobează diverse produse harware și software ce gestionează o mare colecție de baze de date ce conțin informații și servicii de toate tipurile, acestea fiind accesibile de la orice calculator (terminal) conectat la rețea în orice loc și la orice moment.

Comentariu

Mediul *Internet* este cel mai mare sistem informatic din lume, perfect legal și în multe cazuri gratuit. Resursele pe care le oferă sunt imense și ele se schimbă aproape zilnic, așa cum se schimbă tehnologiile informatice.

În România, după 1990, rețeaua *Internet* a cunoscut o expansiune deosebită prin apariția diverselor legături interne și internaționale, constituite în diverse rețele. La început a apărut rețeaua din mediul universitar (nodul central fiind UPB - *Universitatea Politehnică București*) și rețeaua din mediul cercetării (nodul central fiind ICI - *Institutul de Cercetări în Informatică*).

Rețeaua locală UPB este nodul central al rețelei RoEduNet (*The Romanian Education Network*) la care sunt conectate rețele din peste 100 de instituții de învățământ și de cultură din București și din țară. În prezent, majoritatea centrelor universitare din țară au rețele locale conectate la *Internet*. De asemenea, multe școli generale și licee din București și din țară sunt conectate la *Internet*. Majoritatea instituțiilor guvernamentale și neguvernamentale, institute de cercetare și proiectare, marile biblioteci din țară, firme cu diverse activități sunt conectate la *Internet*.

Cererea pentru conectarea la rețeaua *Internet* a determinat apariția furnizorilor de servicii *Internet* atât pentru domeniul public, cât și pentru domeniul comercial.

Furnizori de servicii publice

- RoEduNet (*The Romanian Education Network*);
- SFOS (*Soros Foundation for an Open Society - Fundația Soros pentru o societate deschisă*);
- RNC (*National R&D Network - Rețeaua Națională de Calculatoare pentru Cercetare și Învățământ*).

Furnizori de servicii comerciale

Astăzi, furnizorii de servicii comerciale sunt în număr mare, de aceea vom aminti numai câțiva: *PC-NET Data Network*, *EUnet Romania*, *EuroNET Romania*, *Kappa Servexim*, *IIRUC-RoLink*, *Web Club*, *Dynamic Network Technologies*, *Electronum*.

Evoluția și dezvoltarea Internet-ului

În anul 1968 guvernul S.U.A a dorit interconectarea pe o întindere mare a unui număr mare de calculatoare (de diverse tipuri și dimensiuni), în scopul cooperării dintre universități și departamentele militare ale țării în cadrul unor proiecte de cercetări comune. Din aceste motive a luat ființă agenția ARPA (*Advanced Research Projects Agency*) a cărei activitate s-a concretizat în anul 1970 prin proiectele:

- ARPANET - rețea de calculatoare pentru cercetări în beneficiul guvernului S.U.A.;
- MILNET - rețea de calculatoare pentru cercetări în domeniul militar și de apărare.

În anul 1975 s-a realizat standardizarea unor protocoale (limbaje de comunicare între calculatoarele interconectate) obținându-se standardul TCP/IP (*Transmission Control Protocol/Internet Protocol*) utilizat pentru legarea în rețea a calculatoarelor de la majoritatea universităților americane importante și de la departamentele de apărare din S.U.A. Astfel, a apărut rețeaua de calculatoare DARPA Internet, apoi DARPANET și care a reprezentat nucleul unei gigantice rețele de calculatoare pe care o numim astăzi mediul *Internet*.

După anul 1980, deja au început să se lege la acest nucleu tot mai multe rețele locale, principalele legături *Internet* realizându-se prin organizația NSF (*National Science Foundation*), gestionate de compania ANS (*Advanced Network and Services*) în cooperare cu firmele IBM, MCI și un consorțiu de universități din Michigan. De asemenea, în această perioadă s-a realizat și interconectarea cu rețele de pe continentul european prin intermediul comunicațiilor prin satelit.

Trebuie să precizăm faptul că în anii '80 s-au constituit rețele indepentente care ulterior s-au conectat la rețeaua *Internet* (prin sisteme numite ***gateways*** - porți electronice de legătură) :

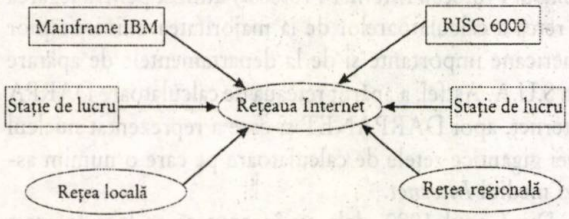
- rețeaua *Usenet*, a luat ființă pentru a conecta calculatoarele Universității Duke cu cele ale Universității din

Carolina de Nord (implementată de *Tom Truscott* și *Jim Ellis* în 1980);
- rețeaua CSNET (*Computer Science Network*), 1980;
- rețeaua NSFNET (*National Science Foundation Network*), 1986;
- rețeaua *FidoNet* care a implementat serviciul *echomail* dezvoltat de *Tom Jennings* în 1984; astăzi, se poate accesa acest serviciu prin WWW;
- rețeaua BITNET (*Because It's Time Network*), rețea mondială formată din universități, institute de cercetare, organizații comerciale; în anul 1989 a fuzionat cu rețeaua CSNET constituind rețeaua CREN (*Corporation for Research and Education Network*);
- rețeaua BIX (*Byte Information eXchange*), rețea creată de revista *Byte* pentru servicii on-line orientate pe probleme hardware;
- rețeaua EARN (*European Academic and Research Network*).

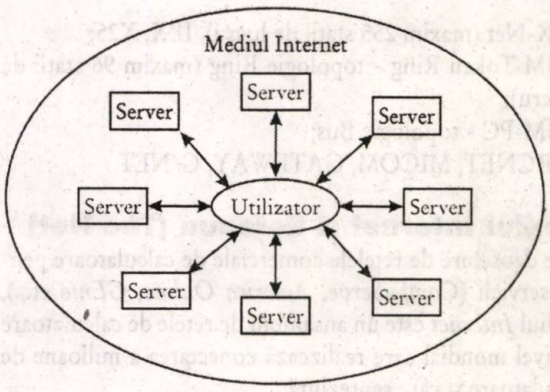
Din anul 1992, administrarea mediului *Internet* este realizată de organizația internațională ISOC (*Internet Society*) care este sponzorizată de membrii comunității *Internet* din S.U.A., Europa și de pe alte continente. Organizația ISOC are următoarele organisme administrative:
- IAB (*Internet Architecture Board*) - se ocupă cu dezvoltarea protocoalelor din rețea;
- IETF (*Internet Engineering Task Force*) - se ocupă cu dezvoltarea specificațiilor pentru standarde;
- IRTF (*Internet Research Task Force*) - se ocupă cu tehnologia interconectării și cu știința informației.

Arhitectura și funcționarea Internet-ului

Din punct de vedere arhitectural, rețeaua *Internet* este constituită dintr-o mare varietate de sisteme de calcul: PC-uri, mainframe-uri, supercalculatoare, rețele de calculatoare LAN, MAN sau WAN.

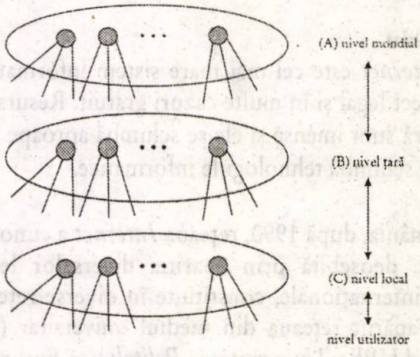


Funcționarea rețelei *Internet* se realizează prin intermediul unor noduri (server-calculator) care oferă cu ajutorul unor protocoale exploatarea eficientă a resurselor mediului *Internet*, noduri ce se numesc furnizori de servicii.



Un utilizator poate accesa serviciile *Internet* dacă are acces la un terminal (calculator - stație de lucru) conectat la rețeaua *Internet*, adică la un nod din rețea. Terminalul se poate afla acasă, la școală, la facultate, la bibliotecă, la serviciu, la un *Internet-café* etc.

Din punct de vedere arhitectural, în sistemul *Internet* sunt recunoscute trei tipuri de noduri:
- noduri de nivel înalt - noduri (clasa A) la nivel mondial la care sunt conectate nodurile continentale (de exemplu, nodul EARN - *European Academic and Research Network*) de la nivelul țărilor;
- noduri continentale - noduri (clasa B) la nivelul unei țări (în România există nodul de la UPB, nodul de la ICI etc.) la care sunt conectate noduri locale ale unor rețele;
- noduri locale - noduri (clasa C) la nivel local (regional) care trebuie să fie recunoscute de un nod continental (în România există noduri locale în București - *Universitatea București*, *Institutul de Matematică*, *ASE*, *Guvern*, *Parlament* etc. - și în marile orațe - Cluj, Iași, Constanța, Craiova, Pitești etc.).

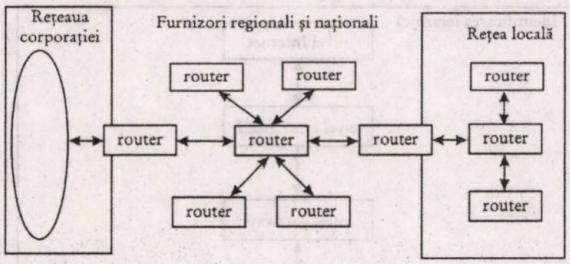


Din punct de vedere funcțional, în sistemul *Internet* sunt operaționale trei tipuri de calculatoare (servere):

Elemente...

Fizic, sistemul *Internet* este un ansamblu de componente hardware (dispozitive de memorare și procesare, echipamente I/O, echipamente de conectare, echipamente de transmisie) pentru memorarea, procesarea, conversia, reprezentarea și comunicarea informațiilor la nivel local, regional sau mondial, iar logic, este un ansamblu de componente software (sisteme de operare - *Unix*, *Linux*, *Windows* etc., programe de control și transmisie, programe de protecție, aplicații pentru căutare, procesare și reprezentare) pentru a oferi utilizatorilor conectați la rețea servicii variate în orice moment și la orice distanță de resursele accesate, pe principiul *client-server*.

- calculator ***router*** (de dirijare) - calculator ce furnizează servicii de dirijare a informațiilor între două noduri care se adresează prin emisie/recepție;
- calculator ***gateway*** (de legătură) - calculator de legătură între nivelurile rețelei *Internet* și care realizează conectarea între două rețele distincte;
- calculator ***host*** (gazdă) - calculator conectat la rețeaua *Internet* pe unul dintre cele patru niveluri (utilizator, local, țară și mondial) de la care se pot cere servicii *Internet*.



Mediul Internet

Un utilizator poate avea acces la serviciile *Internet* numai dacă lucrează la un calculator conectat la un nod din rețeaua *Internet*. Conectarea la un astfel de nod (*server*) se realizează fizic prin intermediul unei firme specializate care poate folosi două metode:

- conectare tip *Network* - cablu coaxial;
- conectare tip *dial-up* - modem, linie telefonică comutată.

În vederea utilizării serviciilor poștei electronice (*e-mail*), cu acordul administratorului nodului respectiv, utilizatorul primește un cont (*nume*) de utilizator (*login name*) și o parolă (*password*) de conectare ce va fi recunoscută în mediul *Internet*. În felul acesta, utilizatorul conectat la *Internet* va avea acces la toate serviciile oferite de *Internet*, inclusiv la aplicațiile mari și complexe din domeniile: științific, învățământ, cercetare, economic, comercial, sportiv, artistic etc.

Accesarea serviciilor mediului *Internet* se realizează de la calculatorul utilizatorului conectat la *Internet*, care trebuie să funcționeze sub unul dintre sistemele *Windows 95/98/NT/2000*, *Linux*, *Unix* etc. și să aibă instalate o serie de programe de navigare precum *Netscape Navigator* sau *Microsoft Internet Explorer*.

În această comunitate de rețele de calculatoare interconectate, pentru realizarea serviciilor oferite, mediul *Internet* are la bază sistemele de operare *Unix* sau *Linux*, sisteme ce oferă facilități moderne pentru comunicarea în rețea (servicii de interconectare):

- Telnet - limbaj pentru comunicarea în rețea; conectarea la distanță;
- E-mail (*Electronic Mail*) - poșta electronică (trimiterea și primirea scrisorilor);
- FTP (*File Transfer Protocol*) - transfer de fișiere;
- HTTP (*HyperText Transfer Protocol*) - transfer de informații aflate la distanță.

Mediul *Internet* utilizează și instrumente (*sisteme hipertext*) concepute special pentru localizarea, căutarea și vizualizarea în rețea, cum ar fi:

- WWW (*World Wide Web* - pânza mondială) - sistem de căutare și vizualizare;
- Gopher - explorare baze de date;
- Archie - căutarea fișierelor în rețeaua *Internet*;
- WAIS (*Wide Area Information Server*) - căutarea și extragerea informațiilor.

Funcțiile principale ale *Internet-ului* pot fi clasificate în:

- localizare și căutare (fișiere, adrese etc.);
- comunicare/transfer (fișiere, scrisori, pachete etc.);
- prelucrare (operații I/O, calcule etc.);
- protecția informațiilor și operațiilor (accesul cu parolă, drepturi de utilizare).

Mediul *Internet* este conceput și construit în jurul conceptului de niveluri de serviciu și reprezintă o rețea de comutare de pachete (PSN - *Packet Switching Network*). Comunicarea (transferul) informațiilor în mediul *Internet* se realizează prin colecții de date numite pachete (*packets*).

Regulile utilizate în mediul *Internet* pentru a decide cum și unde sunt transmise pachetele, formează așa-numitele protocoale de comunicații care constituie software+ul mediului *Internet*. Protocoalele reprezintă pentru comunicații ceea ce reprezintă limbajele de programare pentru prelucrarea informațiilor.

Servicii Internet

E-mail (poștă electronică)

Prin utilizarea comenzilor *Unix/Linux* `mail`, `pine`, `elm` etc. se pot trimite mesaje (scrisori) de poștă electronică spre utilizatori din întreaga lume, se poate vizualiza lista scrisorilor primite în cutia poștală (*mailbox*), se pot recepționa mesaje din întreaga lume.

Rețeaua UseNet

Folosind un cititor de știri (*newsreader*), cum ar fi `rn`, `trn`, `nn` sau `tin`, se pot studia mesaje din grupurile de știri (*newsgroups*) pe orice temă: calculatoare, știință, medicină, economie, sport, muzică etc.

Zone de informare (WWW, Gopher, FTP)

Informațiile sunt stocate în sisteme de fișiere numite zone (*site*) sau gazde (*host*):

- zona *FTP* este un calculator (server) inițializat pentru conectarea în sistem, consultarea și copierea de fișiere;
- zona *Gopher* conține baze de date cu referințe încrucișate despre documente din întrega lume;
- zona *WWW* conține pagini Web care pot fi vizualizate cu programe speciale de consultare (*browser*), cum ar fi *Mosaic*, *Netscape Navigator*, *Internet Explorer*; paginile *Web* pot conține trimiteri la alte zone *WWW* la care există acces.

Mediul *Internet* cuprinde un set de protocoale de rețea care specifică detaliile comunicațiilor între calculatoarele (nodurile) interconectate, împreună cu convențiile de interconectare a rețelelor și de dirijare a informațiilor în rețea.

În prezent, majoritatea rețelelor conectate la *Internet* funcționează folosind interconectarea *TCP/IP* (sistemul de operare *Unix* a fost unul din primele sisteme de operare care au inclus protocolul TCP/IP) care este constituită din protocoalele:

- TCP (*Transmission Control Protocol*);
- IP (*Internet Protocol*);
- UDP (*User Datagram Protocol*).

Observație

Există și protocoale mai vechi, cum ar fi *IPX* (firma *Novell*) și *X25*.

Funcțiile de prelucrare și comunicație ale mediului *Internet* se realizează prin intermediul unor procese cooperante. Această funcție de cooperare este evidențiată în modelul arhitectural al sistemelor teleinformatice, la baza cărora se află conceptul de stratificare ce constă într-o ierarhie de niveluri (straturi) funcționale, fiecare nivel oferind utilizatorilor servicii furnizate de nivelul funcțional inferior.

Modul de funcționare a mediului *Internet* este asemănător activităților poștale care se realizează prin oficii și suboficii poștale. Calculatoarele *router* au rolul suboficiilor poștale care iau decizii asupra modului de dirijare a datelor (pachetelor) la fel cum serviciul poștal direcționeaza scrisorile spre destinație. Un oficiu poștal nu are legătură directă cu toate oficiile poștale dintr-o țară, ci are legătură directă cu unele oficii poștale vecine.

Pentru ca un pachet (împachetare de date) să ajungă la destinație, se utilizează o informație de adresă. Un calculator *router* primește informația de adresă și, pe baza acesteia, decide unde să fie trimis pachetul, stabilind doar "conducta" (*pipe*) optimă pentru ca pachetul să ajungă la destinația indicată de adresa corespunzătoare pachetului.

Regulile utilizate în mediul *Internet* pentru deciziile privind dirijarea pachetelor sunt constituite în protocoale de comunicație.

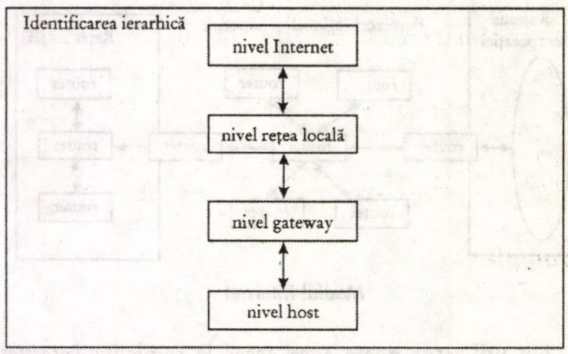
Adrese IP și structura DNS (Domain Name System)

Fiecare calculator conectat la mediul *Internet* este identificat printr-o adresă unică, cunoscută sub numele de *IP Address*, aceasta fiind utilizată la nivelul programelor de prelucrare în rețea. În schimb, la nivelul utilizatorilor cu acces la mediul *Internet*, identificarea calculatoarelor se face printr-un nume de calculator *host* gestionat de sistemul DNS.

Concepția protocolului TCP/IP este implementată astfel încât schema de adresare permite utilizatorilor și programelor din rețea să identifice în mod unic o rețea sau un calculator *host*, atât prin intermediul unei adrese IP, cât și prin intermediul unui nume de calculator *host*.

Operațiile de identificare în mediul *Internet* pot fi clasificate astfel:

- identificare la nivelul programelor de comunicație - adresa IP (*IP Address*);
- identificare la nivelul utilizării - nume de calculator *host* (*DNS Configuration*);
- identificare utilizator - nume de conectare (*login name*).

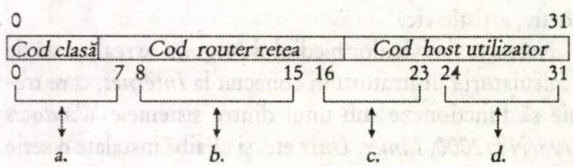


Adrese IP (IP Address)

Definiție

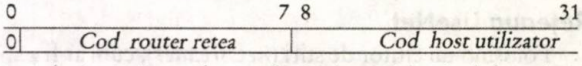
O adresă IP este constituită dintr-un grup de patru numere zecimale separate de caracterul punct (" . "), fiecare număr fiind cuprins între 1 și $256=2^8$ (fiecare număr poate fi reprezentat în binar pe un octet), și anume: pornind de la această structură de numere zecimale, protocolul IP construiește (reprezintă) un cod (adresă) în binar pe un cuvânt de 32 biți (4 octeți) ce va fi constituit din:

- clasa rețelei - clasa A, B sau C;
- *router*-ul rețelei - identificator de rețea;
- *host*-ul utilizatorului - identificator *host*.



Din punct de vedere arhitectural, după cum am precizat, în sistemul *Internet* sunt recunoscute trei tipuri de noduri:

- noduri de nivel înalt - noduri (clasa A) la nivel mondial la care sunt conectate nodurile de la nivelul țărilor:
  - cod clasă - utilizează doar bitul 0 care este 0;
  - cod *router* rețea - utilizează 7 biți;
  - cod *host* utilizator - utilizează 24 biți și prin urmare sunt disponibile 126 coduri de *router*, $2^{24}=16777216$ coduri de host;



- noduri continentale - noduri (clasa B) la nivelul unei țări de care sunt conectate noduri locale ale unor rețele:
  - cod clasă - utilizează biții 0 și 1 care au valoarea 10;

- cod *router* rețea - utilizează 14 biți;
- cod *host* utilizator - utilizează 16 biți și prin urmare sunt disponibile 2^{14}=16384 coduri de *router* și 2^{16} = 65536 coduri de *host*;

0 1	2 15	16 31
1 0	*Cod router retea*	*Cod host utilizator*

• noduri locale - noduri (clasa C) la nivel local (regional) care trebuie să fie recunoscute de un nod continental:
- cod clasă - utilizează biții 0-2 care au valoarea 110;
- cod *router* rețea - utilizează 22 biți;
- cod host utilizator - utilizează 8 biți și prin urmare sunt disponibile 2^{22} = 4194304 coduri de *router*, 2^{8} = 256 coduri de *host*.

0 1 2	3 23	24 31
1 1 0	*Cod router retea*	*Cod host utilizator*

Exemple

1. 1.37.109.4 server de clasa A (nivel înalt) de la *Tilburg University* (Olanda);
2. 163.1.62.3 server de clasa B (nivel țară) pentru conectarea la diverse biblioteci universitare din întreaga lume (*login*: HYTELNET);
3. 141.85.128.1 server UPB;
4. 153.226.100.1 server de clasa B (nivel țară) de la *Facultatea de Chimie* (Panduri);
5. 193.226.51.177 server de clasa C (nivel local) ce reprezintă *gateway* (*gw-chimie*) la *Facultatea de Chimie* din cadrul *Universității București* și la care sunt conectate *host*-urile având adresele IP de la 193.226.51.178 la 193.226.51.190;
5. 192.129.3.33 server de clasa C (nivel local) ce reprezinta un *router* (*math*) la *Facultatea de Matematică* din cadrul *Universității București* unde există mai multe rețele locale.

Adresa IP este utilizată de protocolul IP pentru obținerea în binar a unui cuvânt de memorie (*word*) de 32 biți și care va fi utilizat în operațiile de dirijare a pachetelor (datelor). De exemplu, pentru adresa reprezentată în zecimal 131.15.3.19, cuvântul de memorie care reprezintă forma binară este (se concatenează cei 4 octeți asociați):

1000001	00001111	00000011	00010011
↕	↕	↕	↕
131.	15.	3.	19

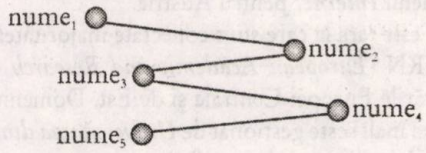
Structura DNS (Domain Name System)

După cum am precizat, fiecare calculator conectat la mediul *Internet* este identificat printr-o adresă unică, cunoscută sub numele de *IP Address*, aceasta fiind utilizată la nivelul programelor de comunicații în rețea. În schimb, la nivelul utilizatorilor cu acces la mediul *Internet*, identificarea calculatoarelor se face printr-un nume de calculator *host* gestionat de structura DNS. Evident, protocolul IP realizează o corespondență bijectivă între adresele IP și numele gestionate de DNS.

Structura DNS realizează administrarea unor nume prin care se acordă diferite responsabilități de grup, fiecare nivel reprezentând un domeniu. Domeniile sunt gestionate prin intermediul unei structuri arborescente care determină o structură ierarhică. În felul acesta, un nume de calculator *host* va reprezenta un drum (cale) în arborele sau subarborele administrat de DNS. Domeniile determină metoda de organizare a calculatoarelor în rețea.

Definiție

Un nume de calculator *host* este constituit din maxim 5 nume de domenii separate de caracterul punct (" . "), ce va reprezenta legătura cu nivelul superior, domeniul din stânga fiind de nivel inferior, iar domeniul cel mai din dreapta având nivelul cel mai înalt, și anume are următoarea reprezentare lexicală: $nume_5.nume_4.nume_3.nume_2.nume_1$, și care reprezintă următorul drum în arborele administrat de DNS:



În prezent, mediul *Internet* consideră $nume_1$ ca domeniu principal, iar celelalte ca subdomenii, această structură de organizare fiind asemănătoare organizării fișierelor în sistemul de operare *Unix*.

Structura ierarhică generată de domenii și subdomenii este definită în funcție de diferite unități de organizare sau de diverse domenii de activitate.

Următoarele domenii sunt recunoscute astăzi ca domenii principale (nivelul înalt):

Com	societăți și companii comerciale;
Edu	organizații educationale (universități, colegii);
Gov	organizații guvernamentale;
Mil	organizații militare (armata, marina);
Int	organizații internationale;
Org	alte organizații;
Net	diverse resurse pentru rețeaua *Internet;*
<țara>	codul de țară.

Codul de țară reprezintă o secvență de două litere, de exemplu:

au	- Australia	*fr*	- Franța
at	- Austria	*hk*	- Hong Kong
bg	- Bulgaria	*hu*	- Ungaria
ca	- Canada	*jp*	- Japonia
de	- Germania	*no*	- Norvegia
pl	- Polonia	*pt*	- Portugalia
ro	- România	*es*	- Spania
se	- Suedia	*yu*	- Iugoslavia

Rețeaua ARPANET (1969) fiind cea mai veche, din care s-a desprins rețeaua MILNET, s-a structurat pe scheletul unor organizații responsabile de diverse domenii. O

organizație care coordonează un domeniu va fi responsabilă de toate subdomeniile ce se definesc în domeniul respectiv.

Astăzi, principalele legături în mediul *Internet* aparțin organizației NSF (*National Science Foundation*), gestionate de compania ANS (*Advanced Network and Services*) în cooperare cu companiile IBM, MCI și un consorțiu de universități din Michigan (SUA).

După ce s-a extins și în alte țări, mediul *Internet* a definit domeniile la nivelul unei țări, ca domenii de nivel înalt.

Comentariu

În prezent, mediul *Internet* este constituit din circa 8500 de rețele conectate, aproape 2,5 milioane de calculatoare, circa 21.000 domenii și milioane de utilizatori utilizează resursele sale.

Exemple

Vă prezentăm în continuare un arbore ierarhic de domenii/subdomenii *Internet* pentru Austria.

Austria este țara la care sunt conectate majoritatea nodurilor EARN (*European Academic and Research Network*) din țările Europei Centrale și de Est. Domeniul de țară (de nivel înalt) este gestionat de *Universitatea din Viena* (*univie*) și conține subdomeniile:

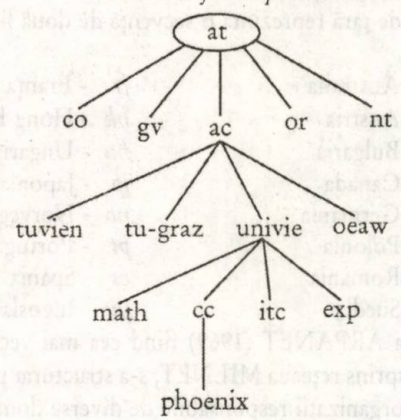
ac	academic;
co	comercial (gestionat de *EUnet Ltd.*);
gv	guvernamental;
or	organizație;
nt	rețele.

Subdomeniul academic *ac* conține, de exemplu:

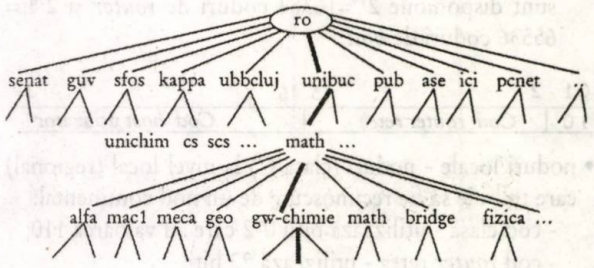
univie	*Universitate din Viena*;
oeaw	*Academia de Științe a Austriei*;
tuvien	*Universitatea Tehnică din Viena*;
tu-graz	*Universitatea Tehnică din Graz.*

În *Universitatea din Viena* există, de exemplu, următoarele subdomenii:

cc	*Centrul de Calcul* (*Computer Center*) (ce conține *phoenix*);
math	*Institute for Mathematics;*
itc	*Institute for Theoretical Chemistry;*
exp	*Institute for Experimental Physics.*



Iată și un exemplu pentru un arbore ierarhic de domenii/subdomenii *Internet* din România.



Concluzie

Conform interconectării în rețeaua *Internet*, un calculator *host* este subordonat din punct de vedere al comunicațiilor altui calculator *host* și subordonează, la rândul său, alte calculatoare *host*. Numele unui calculator *host* reprezintă modul de localizare în structura generată de interconectarea calculatoarelor în rețeaua *Internet*.

Adresa E-mail pentru utilizator

Definiție

Adresa *e-mail* a unui utilizator cu acces la seviciile poștei electronice oferite de rețeaua *Internet* este următoarea construcție lexicală: `numelogin@host.domeniu`, unde:

- *numelogin* este numele (*login name*) declarat de utilizator atunci când i se atribuie accesul la serviciile *e-mail* (asociat cu o *parolă*);
- *host* este numele calculatorului *host* cu rol de *server* (*gateway*) de nivel inferior;
- *domeniu* este drumul (calea) în arborele unui domeniu principal.

Observație

Semnul `"@"` este caracterul care se citește `"at"` (*coadă de maimuță*).

Exemple

1. vlada @math.math.unibuc.ro
2. socmat@math.math.unibuc.ro
3. infodata@theoffice.net
4. office@ginfo.ro
5. mateescu@agora.ro
6. teora@teora.kappa.ro
7. vlada@gw-chimie.math.unibuc.ro

Gestiunea adreselor *Internet* se face de către DDN (*Defense Data Network*), NIC (*Network Information Center*). O adresă se poate obține trimițând un *e-mail* la info@internic.net.

Domnul Marin Vlada este lector la Universitatea București și poate fi contactat la adresa vlada@math.math.unibuc.ro.

Procesoare la 1 GHz

Adrian Monea, Mihai Scorțaru

Ziua de 6 Martie 2000 va rămâne în istoria industriei informatice ca data la care a fost lansat pe piață primul procesor pentru PC-uri care are o frecvență de 1 GHz și funcționează la temperatura camerei. Compania care a atins prima bariera de 1 GHz a fost Advanced Micro Devices, rivalii de la Intel lansând două zile mai târziu propriul procesor care funcționează la o frecvență de 1 GHz.

În cele din urmă, spectaculoasa cursă spre pragul psihologic de 1 GHz a luat sfârșit, protagoniștii principali: trecând linia de sosire aproape simultan. Această întrecere s-a încheiat mult mai repede decât ne-am fi așteptat, inițial *Intel* și *AMD* au anunțat că vor lansa procesoare cu frecvențe de 1 GHz sau mai mult, doar spre sfârșitul anului 2000 sau începutul anului 2001. Se pare că utilizatorii nu au mai avut răbdare și, la presiunea acestora, *Intel* și *AMD* au fost nevoite să lanseze mult mai repede aceste procesoare.

AMD Athlon 1000

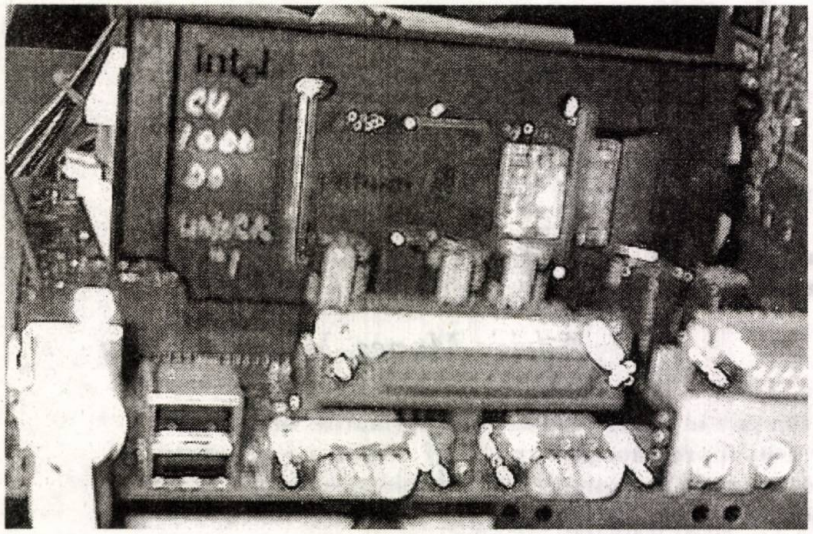
Pe data de 6 Martie 2000, în *Sunnyvale-California*, *AMD* a lansat pe piață primul procesor pentru PC-uri din istorie care funcționează la 1000 MHz și nu necesită dispozitive speciale de răcire. Cu această ocazie, *W.J. Sanders III, chairman* și *chief executive officer* la *AMD*, a declarat că: "Atingerea pragului de 1 GHz a fost de mult timp unul dintre scopurile principale ale producătorilor de procesoare pentru PC-uri. Lansarea unui procesor capabil să execute un miliard de cicluri de ceas pe secundă este, pentru industria noastră, un eveniment echivalent cu depășirea barierei sunetului. La fel cum realizarea lui *Chuck Yeager* a marcat începutul unei noi ere în aviație, procesorul de 1 GHz anticipează o nouă eră în tehnologia informației."

Deoarece se știa că *Intel* va lansa peste câteva zile un *Pentium III* la 1 GHz, *Sanders* a mai spus, "Fiecare școlar știe că *Neil Armstrong* a fost primul om care a călcat pe *Lună*, că *Roger Bannister* a fost primul care a parcurs o milă în mai puțin de patru minute sau că *Edmund Hillary* a fost primul care a escaladat vârful *Everest*. Nimeni nu-și mai amintește cine au fost cei care au realizat aceste lucruri pentru a doua oară."

Procesorul *AMD Athlon 1000* a fost lansat la numai nouă luni după lansarea primului procesor *Athlon* care funcționa la o frecvență de 500 MHz, contrazicând celebra lege a lui *Moore* care prognoza o dublare a performanțelor la fiecare 18 luni.

Primele firme care au lansat sisteme dotate cu procesoare *Athlon* la 1 GHz au fost *Compaq* și *Gateway*, firme care au impus și devansarea datei lansării acestui procesor. Prețul minim al unui sistem dotat cu acest procesor este de 2499$.

La data lansării, un procesor *AMD Athlon 1Ghz* costa 1299$ (pentru comenzi de minimum 1000 de bucăți). Cu acea ocazie *AMD* a prezentat și versiunile *AMD Athlon 950* și *AMD Athlon 900* care aveau prețuri de 999$, respectiv 899$, tot pentru comenzi de minimum 1000 de bucăți.



Intel Pentium III 1000

Replica lui *Intel* la lansarea procesorului *AMD Athlon 1 GHz* nu s-a lăsat mult timp așteptată, procesorul *Intel Pentium III* cu frecvența de 1 GHz fiind lansat pe data de 8 Martie 2000. Cu această ocazie, ***George Alfs***, purtătorul de cuvânt al companiei *Intel*, a afirmat că acest procesor este destinat doar celor care îi pot aprecia cu adevărat calitățile, cum ar fi utilizatorii entuziaști și adevărații gameri. În al doilea trimestru al anului 2000 numărul procesoarelor disponibile pe piață va fi limitat.

Primele firme care folosesc procesorul produs de *Intel* în sistemele lor sunt *Dell*, *HP* și *IBM*. Prețul minim al unui astfel de sistem este de 3499$.

Este evident faptul că *Intel* nu era pregătită pentru o astfel de lansare, compania fiind nevoită să sară direct de la 800 la 1000 MHz, fără ca pe piață să existe versiuni intermediare.. Procesorul *Pentium III 1 GHz* va costa 990$ pentru comenzi de minimum 1000 de bucăți. După două săptămâni, pe data de 20 martie 2000, *Intel* a lansat procesoarele *Pentium III* la 850 și 866 MHz.

Athlon vs. Pentium III

Este evident că, din punct de vedere al prețului, *Pentium III 1GHz* este mult mai convenabil decât *AMD Athlon 1 GHz* fiind cu aproape 25% mai ieftin. Totuși, datorită numărului mic de procesoare *Pentium III 1GHz* disponibile, este destul de dificil pentru cineva să cumpere un astfel de procesor. Un simpatizant *AMD* spunea: "Ați putea cumpăra procesoare de 1 GHz și de la *Intel* dacă ar exista în lume mai mult de 1000 de bucăți." Au fost și comentarii mai hazlii, unii amintindu-și de următoarea glumă:

O doamnă intră într-un magazin și cere un kilogram de mere. Vânzătorul îi aduce merele și spune "99¢ vă rog". Doamna foarte surprinsă răspunde "dar la super-market merele sunt numai 79¢". Vânzătorul, mirat, o întreabă: "Dar atunci de ce nu ați mers să cumpărați de la super-market?" Foarte serioasă doamna răspunde: "Am fost și acolo, dar mi-au spus că s-a epuizat stocul de mere." Zâmbind, vânzătorul spune: "Stimată doamnă, atunci când nu am mere în stoc, la mine kilogramul costă 29¢."

Realitatea este că, într-adevăr, numărul procesoarelor *Pentium III 1 GHz* disponibile este foarte mic. Între timp, există procesoare *AMD Athlon 1 GHz* care pot fi achiziționate și separat, fără a fi nevoie să cumpărați un întreg sistem.

Din punct de vedere al performanței, s-ar părea că *Pentium III 1000* este superior unui *AMD Athlon 1000* în aplicațiile pentru birou. Aceasta se datorează faptului că procesoarele *Pentium III "Coppermine"* au un cache de 256K integrat pe procesor și care rulează la frecvența acestuia, în timp ce procesoarele *Athlon*, deși au un cache de 512K, acesta nu este integrat în procesor și rulează la mai puțin de jumătate din frecvența procesorului. Totuși procesoarele *Athlon* sunt net superioare în aplicațiile științifice care necesită multe calcule în virgulă mobilă. (Probabil cea mai cunoscută astfel de aplicație este *Quake III*.)

În curând *AMD* va introduce o nouă variantă *Athlon* (nume de cod *Thunderbird*) care va elimina handicapul cache-ului, folosind o tehnologie asemănătoare celei de la *Intel*. În plus, *Athlon* are o arhitectură internă mult mai avansată, care poate fi mult îmbunătățită, în timp ce, *Pentium III* folosește aceeași arhitectură (mult îmbunătățită) a procesorului *Pentium PRO*, lansat în 1994 și care se află la capătul posibilităților de evoluție. *Intel* va lansa o nouă arhitectură pentru procesoarele sale (nume de cod *Wilamette*), probabil numai în cursul anului 2001. Se pare că, pe termen scurt, *AMD* va fi lider în "războiul procesoarelor".

O nouă "bătălie" în acest război va începe o dată cu lansarea procesoarelor pe 64 de biți. *Intel* și-a denumit viitorul procesor *Itanium* și îl va lansa probabil în toamna acestui an. *AMD* va lansa propriul procesor pe 64 de biți doar în vara anului 2001; acesta este deocamdată cunoscut doar sub numele de cod *Sledgehammer*. Se pare că, și în acest caz, *AMD* este în avantaj deoarece *Sledgehammer* oferă suport nativ pentru aplicațiile actuale pe 32 de biți care vor rula chiar mai rapid decât cu procesoarele actuale, la fel cum procesoarele *386* pe 32 de biți rulează mai rapid aplicații pe 16 biți proiectate pentru procesoarele *8086* și *286*. Procesoarele *Itanium* vor putea rula aplicații pe 32 de biți, dar nu nativ, ci cu emulare, ceea ce aduce o diminuare severă a performanțelor. Se pare că istoria se repetă și poate că nu puțini își amintesc de *Intel Pentium PRO*, procesor foarte scump la vremea sa, cu performanțe excelente pe 32 de biți care rula aplicații pe 16 biți mult mai încet decât cele pe 32 de biți. De fapt, cele două firme adoptă poziții total diferite: *Intel* susține că utilizatorii vor dori să treacă cât mai repede de la aplicații pe 64 de biți, în timp ce

AMD consideră că aceștia vor dori să ruleze în continuare și aplicații pe 32 de biți.

Comentarii

Unii specialiști din domeniul IT au comparat comportamentul companiilor *Intel* și *AMD* cu un joc de copii. Ei sunt de părere că încă nu era nevoie de un procesor la 1 GHz și că piața nu este încă pregătită pentru o asemenea lansare. Ei susțin că era normal ca *AMD* să lanseze progresiv procesoare *Athlon* la 900, 950 și abia apoi la 1000 MHz, în timp ce *Intel* ar fi trebuit să lanseze procesoare *Pentium III* la 866 și 933 MHz (eventual și la 900 și 950 MHz) înainte de a atinge pragul de 1000 MHz. Totuși, a fost apreciat faptul că *AMD* a umplut golul care s-ar fi creat prin saltul brusc de la 850 la 1000 MHz. Compania *Intel* a fost criticată pentru faptul că, momentan, nu există versiuni intermediare între *Pentium III 800* și *Pentium III 1000*.

Singurul câștig al majorității utilizatorilor este acela că, pentru a putea vinde procesoare de 1 GHz, *Intel* și *AMD* au fost nevoite să reducă prețurile celorlalte procesoare pentru a le mai putea vinde. *AMD* a dat o nouă lovitură rivalei *Intel,* coborând prețul procesorului *AMD Athlon 500* astfel încât să fie comparabil cu prețul unui procesor *Intel Celeron 500*, performanțele procesorului *Athlon* fiind mult superioare.

Din păcate *AMD* se confruntă cu problema plăcilor de bază compatibile cu procesoarele *Athlon* care sunt foarte puține și mult mai scumpe decât cele destinate procesoarelor *Intel*. Situația pare să se schimbe, deoarece, la *CeBit 2000* peste 20 de producători de plăci de bază au prezentat soluții pentru platforma *Athlon* bazate pe noul *chipset VIA KX133*. Actualele plăci de bază folosesc *chipset*-ul *Irongate 750* sau *751* produse de *AMD*; problema acestor *chipset*-uri este aceea că sunt departe de a putea fi numite moderne deoarece nu suportă ultimele standarde *AGP 4x* și *UltraDMA 66*. În plus, majoritatea plăcilor de bază bazate pe acest *chipset* folosesc *design*-ul de referință (engl. ***reference design***) de la *AMD* care este un *design* pe 6 straturi, motiv pentru care prețul plăcilor de bază crește considerabil.

Plăcile bazate pe *chipset*-ul de la *VIA* folosesc un *design* de referință pe 4 straturi ceea ce va duce la o diminuare a costurilor de producție. De asemenea, este oferit suport pentru standardele *AGP 4x* și *UltraDMA 66*. În același timp noul *chipset* elimină unele probleme de incompatibilitate (de exemplu, cu unele plăci video bazate pe procesorul *GeForce 256* de la *nVidia*) raportate de utilizatori. Reducerea costului este foarte importantă, deoarece, marii producători de plăci de bază (cum ar fi *Abit*) nu au lansat pe piață plăci de bază cu *chipset*-ul *Irongate* pentru că profiturile nu ar fi fost conform așteptărilor.

Un ultim aspect este acela că mulți producători de plăci de bază au preferat să nu ofere suport pentru platforma *Athlon* datorită presiunilor exercitate de *Intel*. De exemplu, *Intel* a redus artificial numărul de chipset-uri *BX* disponibile și mulți producători (care au avut curajul să producă plăci de bază pentru procesoarele *Athlon*) nu au mai putut să satisfacă cererile de plăci de bază pentru procesoare *Intel* suferind pierderi importante. Alți mari producători au produs plăci de bază pentru *Athlon* dar, oficial, nu au recunoscut imediat acest lucru. De exemplu, firma *ASUS* a produs placa de bază *K7M* una dintre cel mai perfomante de pe piață, dar, timp de 3 luni, pe *site*-ul *Web* al companiei nu exista nici o informație referitoare la acest produs. (Placa era livrată într-o cutie complet albă, iar update-urile pentru *BIOS* erau disponibile pe un site *FTP* din Germania și nu pe *site*-ul oficial al firmei *ASUS*.)

Concluzii

Una dintre cele mai importante bătălii ale "războiului procesoarelor" s-a încheiat cu victoria celor de la *AMD*. Rămâne de văzut cum vor evolua evenimentele în viitorul apropiat. Până acum un nou procesor lansat trebuia, în principiu, să fie cu 33, 50 sau 66 MHz mai rapid decât versiunea anterioară. Acum ar fi penibil să auzim că *AMD* lansează procesorul *Athlon* la 1,05 GHz sau că *Intel* lansează *Pentium III* la 1,066 GHz. Probabil că o creștere acceptabilă va fi de 100 MHz, dar unii cred că următoarele versiuni ar trebui să realizeze un salt de 500 MHz, deci curând vom avea procesoare de 1,5 GHz. Acest lucru pare destul de puțin probabil, deoarece ar însemna că vom ajunge la 2 GHz în foarte scurt timp (după două versiuni), deci performanțele procesoarelor se vor dubla din nou foarte repede.

Adrian Monea și Mihai Scorțaru sunt studenți în anul II la Universitatea Tehnică din Cluj. Pot fi contactați prin e-mail la adresele amonea@asterix.obs.utcluj.ro, *respectiv* smihai@asterix.obs.utcluj.ro.

VIRUȘI

Flaviu Cibu, Adrian Bugariu

"Dacă nu există nici un motiv ca ceva să nu existe, atunci există în mod obligatoriu" (Legile lui Murphy) sau mai simplu "dacă există un motiv ca ceva să existe, atunci există în mod obligatoriu". Această lege se aplică și în cazul virușilor informatici, "derivați" din virușii biologici (au aceleași funcții) care se supun de asemenea acestei legi. Existența virușilor biologici este de necontestat precum și cea a virușilor informatici, fiind o realitate nefastă pentru oameni. Ei sunt cei care pot face posibilă pierderea de date importante, de timp și chiar distrugerea calculatoarelor, utilizatorul asistând de multe ori neputincios în fața unui asemenea atac. Ca răspuns la aceste atacuri s-au realizat antivirușii, programe care încearcă să imunizeze sistemele informatice.

Conceptul de virus informatic a apărut încă din anul 1972 într-o carte science-fiction *When Charlie was one* scrisă de *D. Gerrold*. La centrul de calcul de la *University of Carolina*, în data de 3 noiembrie 1983, ***Frederick Cohen*** a demonstrat conceptul atât de convingător încât i s-a retras dreptul de a mai utiliza calculatoarele respective. Experții consideră că în prezent se cunosc aproximativ 50000 de viruși.

Virușii informatici sunt programe de dimensiuni mici care au proprietatea de a se autoreplica inserându-se în fișiere sau în zonele sistem ale discurilor flexibile fără acordul utilizatorului și au în majoritatea cazurilor o activitate distructivă.

Utilizatorul este, în majoritatea cazurilor, cel care determină infectarea. În trecut singurul mod de infectare era prin intermediul dischetelor. În prezent interconectivitatea calculatoarelor, care a ajuns la un grad foarte mare prin răspândirea accentuată a rețelelor și prin apariția *Internet*-ului, la acestea adăugându-se creșterea numărului de calculatoare deci creșterea numărului de surse potențiale de infectare, face ca virușii să cunoască o viteză de propagare foarte mare iar neutilizarea unui *antivirus* performant poate avea consecințe grave.

Internet-ul se dezvoltă necontenit și tinde să devină omniprezent. Funcția de bază a *Internet*-ului este de a facilita transmiterea și schimbul rapid de informații. Astfel el devine un "vehicul uriaș" de transmitere a virușilor. Dar *Internet*-ul are mai multe protocoale de transmitere a datelor și fiecare virus este specializat într-unul din aceste protocoale, dar nici existența unui virus care să se propage prin mai multe protocoale nu poate fi considerată imposibilă, totul depinzând de inteligența creatorilor de viruși, numiți ***virogeni***.

Virușii pot fi transmiși și prin *e-mail*. Ei pot veni ca fișiere infectate atașate *e-mail*-ului sau pot fi găsiți chiar în mesajul *e-mail* dacă se folosește formatul HTML. Aceasta este cea mai frecvent utilizată cale de infectare, dar mai există și protocolul de știri (NNTP) unde utilizatorul se poate înscrie la un server de știri de unde va primi periodic *e-mail*-uri, care pot fi virusate. Un alt protocol este acela de încărcare de fișiere (FTP). Virușii mai pot fi întâlniți și în paginile *Web* care conțin text și imagini dar și applet-uri *Java* și controale *ActiveX*, care pot fi mijloace de infectare.

Clasificarea virușilor

Există mai multe tipuri de viruși:

Virus de boot

Infectează *boot record*-ul unui harddisc sau al unei dischete. *Boot record*-ul harddisc-urilor se referă la MBR aflat pe prima pistă a harddisc-ului și la sectorul de boot aflat la începutul fiecărei partiții conținând codul necesar lansării sistemului de operare. Orice dischetă formatată are un sector de *boot* care poate conține codul necesar lansării sistemului de operare, acesta fiind o dischetă bootabilă. Dischetele nebootabile conțin în sectorul de *boot* doar textul *Non-system disk or disk error*. Virușii de *boot* se pot împărți în viruși de MBR și în viruși de sistem.

Un virus de *boot* înlocuiește zona de sistem cu o copie a sa după ce *boot record*-ul original a fost copiat în alt loc sau a fost pur și simplu suprascris. Există însă și viruși care infectează MBR-ul rezumându-se la a se modifica zece bytes. În momentul în care se se încarcă sistemul de pe o dischetă infectată, virusul se va lansa în execu-

ție. El se va încărca în memorie și va executa codul *boot record*-ului original și totul va părea normal. Dar el rămâne de cele mai multe ori rezident în memorie și va infecta prima dată *boot record*-ul harddisc-ului pentru a fi sigur că se va lansa la o nouă încărcare a sistemului. După aceasta, orice accesare a unui harddisc sau a unei dischete va fi interceptată de virus pentru a asigura propagarea sa. Înainte de a infecta, el verifică dacă *boot record*-ul nu a fost deja infectat, și dacă nu, acesta va fi infectat.

Virușii de *boot* sunt capabili să infecteze un calculator, indiferent de sistemul de operare: sisteme *DOS*: *MS-DOS*, *Windows 3.x*, *Windows 9x*, cât și sisteme non-*DOS*: *UNIX*, *Linux*, *NetWare*, *Windows NT/2000* dar acestea de obicei nu se vor mai putea lansa. Virușii infectează, în special, MBR-ul harddisc-urilor. Deoarece mărimea MBR-ului este de doar 512 bytes, virușii cu un cod mare își vor ascunde codul în afara MBR-ului existând patru posibilități de infectare.

Unii viruși folosesc, pentru a se ascunde, zona dintre primul sector al primei piste (MBR) și primul sector al celei de a doua piste de pe harddisc, această zonă fiind ignorată de sistemele de operare respective, deoarece sectoarele sunt ascunse.

Mulți viruși creează sectoare defecte pe disc unde își vor copia propriul cod.

O altă posibilitate este ca virușii să își ascundă codul în zona rezervată directorului principal.

O soluție ingenioasă este crearea de către virus a unei piste suplimentare pe disc, pistă în care își va copia codul.

Întrucât există mai multe tipuri de dischete, numai o serie mică de viruși sunt capabili să infecteze toate aceste dischete. Majoritatea virușilor de *boot* sunt creați pentru a utiliza dischetele de 1,44 megabytes. Prezența unor asemenea viruși poate fi ușor observată dacă sunt folosite dischete protejate la scriere. De fiecare dată când este citită discheta, virușii vor încerca să scrie pe dischetă și zgomotul produs de unitate va fi mai puternic.

Cea mai bună metodă de a preveni infectarea dischetelor este de a folosi protecția acestora dacă se dorește numai citirea lor. De multe ori calculatoarele sunt infectate pentru că sistemul se încarcă folosind o dischetă virusată sau sunt infectate accidental din cauza unei dischete uitate în unitate. Acest lucru poate fi prevenit prin schimbarea opțiunii de încărcare din *BIOS Setup* astfel încât calculatorul să încerce încărcarea sistemului mai întâi de pe harddisc și apoi de pe dischetă.

În cazul dischetelor, dezinfectarea se realizează cu ajutorul unui antivirus sau prin formatare. Dacă harddisc-urile sunt infectate se deviruseaza cel mai sigur cu un antivirus.

Dacă virusul a infectat MBR-ul, se poate folosi și comanda DOS `fdisk /mbr` care reface MBR-ul, însă dacă virusul mută tabela de partiție în alt loc, comanda `fdisk/mbr` va șterge virusul, dar sistemul nu va mai putea fi încărcat pentru că tabela de partiții nu poate fi găsită. Calculatorul va afișa mesajul de eroare DOS `Invalid drive specification`, toate partițiile fiind pierdute. Este necesară repartiționarea harddisc-ului și formatarea sa distrugându-se toate datele. Dacă este infectat sectorul de *boot* se poate utiliza comanda `sys c:` executată de pe o dischetă curată, care va transfera sistemul de pe dischetă pe harddisc.

Virus de program

Infectează fișierele executabile cu extensia `.exe`, `.com` și fișiere *overlay* care au extensia standard `.ovl` sau `.ovr` care se află pe un suport fizic neprotejat la scriere. Odată lansat, un fișier infectat se va executa, și virusul se va propaga găsind noi gazde. Unii viruși care nu sunt de calitate pot infecta și alte fișiere, criteriul fiind existența caracterelor `MZ` care apar la începutul fișierelor executabile. Virușii de program se inserează în general la sfârșitul fișierului, iar la începutul programului adaugă o instrucțiune de salt spre codul virusului iar acesta conține o instrucțiune de salt la începutul programului original. Un alt tip de infectare este cel al faimosului virus *CIH*. În fișierele executabile există zone care nu conțin nici un cod, fiind notate cu 0 în format hexazecimal, deci ele nu sunt utilizate. Virusul *CIH* se distribuie în aceste zone libere care, însumate, trebuie să ofere un spațiu suficient pentru virus, în cazul nostru 1024 bytes. Astfel dimensiunea fișierului nu crește ca în primul caz. Virusul *CIH* face parte din noua generație de viruși de program concepuți pentru mediul *Windows* (*Windows '95* și *Windows '98*), primul astfel de virus fiind *Bizatch*, apărut în ianuarie 1996. Dar există și viruși care pur și simplu suprascriu fișierul executabil care este distrus și există (culmea!) posibilitatea ca dimensiunea fișierului să rămână aceeași. Aceștia sunt numiți viruși de *suprascriere* sau *overwrite*, care însă sunt foarte rapid observați de utilizator deoarece programele nu se vor mai executa. Majoritatea virușilor marchează întru-un fel fișierul infectat pentru a nu crește dimensiunea programelor și a fi mai greu de detectat. În majoritatea cazurilor virușii caută în fișier o secvență din codul lor, dar sunt și cazuri în care este schimbată data modificării fișierului cu o anumită valoare. Sunt și viruși care nu testează dacă fișierul a fost deja infectat și îl infectează în continuare, fiind posibil ca fișierul să atingă dimensiuni foarte mari dacă este accesat periodic.

Clasa virușilor de program se împarte în viruși *rezidenți* și viruși *nerezidenți* în memorie. Virușii rezidenți sunt virușii care o dată lansați în execuție vor rămâne tot timpul în memorie și vor infecta orice program care va fi executat, iar cu cât utilizatorul va executa mai multe programe cu atât mai mult virusul se va răspândi. Cei mai mulți viruși se autodetectează în memorie pentru a nu exista mai multe copii active în același timp. Dar sunt și viruși care nu fac acest lucru, astfel reducându-se mult memoria liberă și odată cu ea viteza calculatorului, ajungându-se la imposibilitatea de a mai rula programe, fapt ce va fi sesizat imediat de utilizator. Virușii nerezidenți se vor executa și vor încerca să

infecteze alte fișiere aflate de obicei în directorul curent și în directoarele la care face referință variabila de mediu PATH, după care se vor șterge din memorie, fiind nedetectabili iar viteza de operare a calculatorului nefiind afectată. Acest tip de viruși mai sunt numiți și viruși *Direct Action*.

O categorie de viruși sunt cei de tip bombă logică. Acești viruși se vor multiplica până când o condiție va fi satisfăcută și în acel moment se va declanșa "bomba". Condițiile puse sunt variabile. Des folosite sunt cele de dată și oră. Alți viruși se declanșează dacă au infectat de un anumit număr de ori, cum este cazul renumitului virus *One Half* care acționează dacă s-au criptat jumătate din fișierele aflate pe disc. Unii viruși chiar chestionează utilizatorii și în caz de răspuns greșit este pornit procesul de distrugere. Dar efectele pot varia de la mesaje și desene amuzante până la pierderea unor date importante și culminând cu distrugerea hardisc-urilor și a calculatorului căruia i-a fost șters BIOS-ul. Împotriva acestor viruși cea mai bună metodă este scanarea cu un antivirus performant al oricărui fișier copiat pe harddisc.

Virus multipartit

Este un hibrid între cel puțin alte două tipuri de viruși. Acești viruși pot infecta mai multe medii: fișiere executabile, MBR-ul, sectorul de *boot*, tabele de alocare a fișierelor, directoare, asigurându-le șanse de supraviețuire mai mari. Cei mai întâlniți sunt virușii de *boot* și program cum este cazul virusului *Kaczor*. Chiar dacă va fi eliminat din fișierele executabile, el va continua să infecteze până când va fi devirusat și MBR-ul.

Virus însoțitor

Speculează o anumită caracteristică a sistemului DOS prin care, dacă două programe, unul având extensia .exe și celălalt extensia .com, au același nume, atunci se execută mai întâi fișierul .com. Deci acești viruși se aplică doar la fișierele executabile cu extensia .exe creând un fișier cu extensia .com care conține virusul. El se lansează în execuție doar în momentul în care este executat un fișier fără a i se preciza extensia. Fiind dată comanda de lansare, se va executa fișierul cu extensia .com și apoi se va executa fișierul cu extensia .exe, totul derulându-se normal. Dezinfecția unui asemena virus este simplă: se încarcă sistemul de pe o dischetă curată după care se șterg fișierele .com care au același nume cu fișierele .exe. De cele mai multe ori fișierele ce conțin virusul sunt ascunse, necesitând modificarea atributelor acestora.

Retrovirus

Acest tip de viruși acționează împotriva antivirușilor. Ei pot căuta un antivirus pe harddisc și îl pot șterge. Un virus activ poate identifica lansarea în execuție a unui antivirus. Ca răspuns, el nu permite lansarea antivirusului sau chiar blochează calculatorul și utilizatorul crede că antivirusul a produs crash-ul. Alți viruși încearcă să limiteze capacitatea de detecție a antivirușilor prin modificarea antivirusului, a unor fișiere ale acestuia. Antivirusul se va comporta normal dar numărul virușilor recunoscuți va fi foarte mic sau chiar zero.

Virușii pot afecta în principal bazele de date despre viruși și fișierele de control. Virusul *Zarma* este capabil să dezactiveze antivirușii *VSafe*, *VDefend* și *Vwatch*. Virusul *Peach* atacă antivirusul *Central Point* prin ștergerea fișierului cu semnăturile virușilor. Neverificând existența fișierului cu amprente, antivirusul se execută fără probleme, însă nu va recunoaște nici un virus. Virusul *GoldBug* șterge toate fișierele Chklist conținute în director, ocolind astfel protecția sumei de control a antivirușilor *Central Point* (CPAV) și *Microsoft* (MSAV). Există și viruși care șterg fișierele sumă de control ale antivirusului TBAV - *ThunderByte AntiVirus*. Un astfel de virus poate în cazul renumitului antivirus AVP - *AntiViral Toolkit Pro* să modifice fișierul avp.set care conține numele bazelor de date cu semnături. Pentru a rula corect, este necesară prezența doar a numelui kernel.avp, iar antivirusul va recunoaște aproximativ 200 de viruși din totalul de peste 30000.

Virus de legătură

Este un tip de virus deosebit deoarece el nu modifică conținutul fișierelor infectate ci modifică structura directorului, mai exact intrările din tabela director corespunzătoare fișierelor executabile. Virușii leagă primul pointer de cluster al intrării de director corespunzătoare fișierelor executabile, de un alt cluster unde se găsește corpul virusului. Lansat în execuție virusul va încărca fișierul executabil citind valoarea cluster-ului de start al fișierului care este stocată în alt loc. Din această categorie pot fi amintiți virușii: *Dir II.Byway* și *DIR II*. Acesta poate fi eliminat prin redenumirea tuturor executabilelor în fișiere cu extensii non-executabile cât timp virusul este activ iar după dezinfectare extensiile fișierelor trebuie reschimbate cum au fost inițial.

În funcție de locul de rezidență virușii prezentați mai sus se clasifică în viruși rezidenți în: blocul de control al memoriei (MCB), *buffer-e DOS*, memoria *Low*, *High*, extinsă, expandată video și în tabela de întreruperi.

O altă clasificare poate fi făcută după modul de infectare. Virușii infectează folosind comenzi *DOS* FindFirst și FindNext sau interceptând comenzi *DOS* ce accesează discul: FindFirst, FindNext, deschidere de fișier, închidere de fișier, citire de fișier.

Un criteriu în clasificarea virușilor este modul de răspândire. Virușii pot fi considerați infectori rapizi sau infectori latenți. De cele mai multe ori infectorii rapizi produc o adevărată epidemie care conduce la diminuarea performanțelor calculatorului, și deci alarmarea utilizatorului.

Virușii pot fi clasificați și după dimensiunea lor. Majoritatea virușilor au o dimensiune cuprinsă între 1500 și 3000 de bytes dar există și microviruși care pot avea o dimensiune de minim 22 bytes cum este cazul virușilor: *Trivial* - 22 bytes, *84* - 84 bytes, *Tiny DI* - 94 bytes, *AT* - 108 bytes,

Smallv - 115, *Anto* - 129 bytes, *Tiny* - 133 bytes, *Toby* și *STSV* - 200 bytes. Caracterstica acestor viruși este că ei doar se răspândesc, dar de obicei suprascriu fișierele infectate. Există și viruși cu dimensiuni cuprinse între 3000 și 5000 bytes, din acestă categorie făcând parte viruși foarte periculoși cum sunt: *Bizatch* - 3192 bytes, *One Half* - 3544 bytes, *Die_Hard* - 4000 bytes, *Frodo* - 4000, bytes *AntiCAD* - 4096 bytes, *Bomber* - 4096 bytes, *Kaczor* - 4444 bytes, *Neuroquila* - 4644-4675 bytes. Virușii cu dimensiune mai mare sunt în general viruși creați în limbaje de nivel înalt: *C*, *Pascal*, *Basic*: *Animus* - 7392 bytes, *Wonder* - 7424 bytes, *Pascal 7808* - 7808 bytes, *Worm 16580* - 16580 bytes, *Ehhehe* - 31107 bytes. Foarte mulți viruși produc diferite efecte sonore și video, uneori distractive, dar și înspăimântătoare (de exemplu, `format...`). Efectele sonore pornesc de la un singur *beep* și ajung la râsete sau chiar la melodii (de exemplu, *Yankee Doodle*). Efectele video ale virușilor sunt foarte variate, de la simplu text afișat, la diferite desene, până la efecte deosebite. În general textele din interiorul virusului sau cele afișate, conțin numele virusului și/sau pseudonimul autorului virusului și un mesaj care precizează infectarea calculatorului.

Virus de macro

Acest tip de virus este în prezent cel mai răspândit. Procesoarele de text ca și alte tipuri de programe au evoluat foarte mult, ajungându-se la implementarea unui "limbaj" care să fie folosit la automatizarea unor task-uri complexe sau repetitive în interiorul documentelor. Astfel de limbaje *macro* se întâlnesc la procesoarele de text *Ami Pro*, *WordPerfect* și la pachetul *Microsoft Office*. Virușii de *macro* sunt virușii scriși în limbajul *macro* al unui program. Ei utilizează limbaje puternice care le permit să modifice (infecteze) și să creeze noi documente conținând viruși de *macro*. Acești viruși sunt întâlniți cel mai frecvent în *Word* și în *Excel*. *Macro*-urile conțin instrucțiuni în *WordBasic* (în *Word 5.0* și *Word 6.0*) sau *VisualBasic* (în *Word '97* și *Word 2000*) ce pot fi utilizate în întreaga suită *Office*, atât pe platforme PC cât și Mac. Mijloacele de infectare sunt fișierele șablon (*template*-uri), care pot conține *macro*-uri. Problema este că orice document se bazează pe un template sistem ale cărui macro-uri vor fi interpretate și executate automat de programul gazdă. De asemenea și documentele pot conține *macro*-uri care să se autoexecute. În cazul aplicației *Word*, o dată deschis un document infectat, virusul se va autolansa și va infecta șablonul global `Normal.dot` sau va crea un fișier în directorul `Startup` care va fi executat la toate sesiunile de lucru. La crearea sau la deschiderea unui document, virusul activat va insera în document propriul său macro folosindu-se de *macro*-urile *Auto* și astfel el își asigură propagarea. Virușii de *macro* se răspândesc cu rapiditate deoarece schimburile de documente sunt mult mai frecvente decât schimburile de programe executabile. *Word '97* permite dezactivarea autoexecuției *macro*-urilor, astfel virușii devenind inofensivi. Dar acestă opțiune poate fi dezactivată chiar de un *macro*. Mulți viruși creați în *Word 6.0* supraviețuiesc și în *Word '97*, fiind convertiți automat în comenzi *VisualBasic*, însă nu și invers. Primul virus *macro* de *Word* a apărut în 1995 și se numește *Concept* dar nu produce daune. În *Excel*, ca și în *Word*, există *macro*-uri *Auto*, care se autoexecută la lansarea mediului. Virușii *Excel* se autosalvează în directorul de start al aplicației *Excel*, `XLStart`, în fișierul `Personal.xls`, iar la următoarea sesiune de lucru ei se vor activa automat. Se cunosc viruși de *macro* care pot distruge toate informațiile de pe harddisc. Ei pot insera comenzi *DOS* cu parametri redirecționați în fișierele de start `Autoexec.bat` și `Config.sys`. Există viruși care pot asambla fișiere executabile. Un astfel de virus asamblează cu ajutorul comenzii `Debug` un fișier `.com` care poate distruge datele de pe disc. Virușii de *macro* devin din ce în ce mai periculoși, acest lucru fiind facilitat și de puternicul limbaj *VisualBasic*. Un virus de macro poate accesa orice aplicație *Office*. Cel mai cunoscut virus de acest tip este *Melissa* pentru că este primul exemplu de succes al unui virus care se răspândește activ prin *Internet*. El se transmite primelor 50 de adrese de mail, pe care le găsește în agenda *Outlook*. Primul și cel mai răspândit virus de *Excel* este *Laroux* dar ca și virusul de *Word Concept* nu are efecte dăunătoare. Nici aplicațiile *Access* și *PowerPoint* nu au rămas imune la viruși. Primul virus *Access* se numește *AccessiV* și se multiplică doar în versiunea Access '97 în limba engleză.

Tehnici evazive

Pentru a nu fi detectați de antiviruși, creatorii de viruși au dezvoltat mai multe tehnici complexe cu care au dotat virușii.

Stealth

Aceasta este o tehnică utilizată de virușii rezidenți. După cum le spune și numele acești viruși încearcă să se ascundă pentru a nu fi detectați. În momentul în care va fi accesat un fișier, virusul va raporta dimensiunea originală. El interceptează vectorii de întrerupere utilizați în citirea datelor de pe disc și îi modifică, astfel oferind informații false oricărui program care citește fișiere infectate. O serie de viruși, înainte de accesul la fișierul infectat, se vor autoextrage iar fișierul va fi unul nevirusat. Dar la închiderea fișierului virusul va reinfecta fișierul, astfel virusul nu va fi detectat de scanner-ele antivirușilor. Această tehnică este utilizată și de virușii de *boot* care nu vor fi detectați în *boot records* când ei sunt rezidenți în memorie. În concluzie, un virus *stealth* poate falsifica orice citire a discului. Folosind un mod de ascundere ingenios, unii viruși interceptează orice încărcare de program și, dacă găsesc în memorie caractere `'A'`, `'V'` sau `'AV'`, ei eliberează memoria pentru a nu fi detectați de scanner-ele de memorie ale antivirușilor. Același lucru se poate spune și la infectarea fișierelor. Virușii verifică mai întâi dacă fișierele exe-

cutabile nu sunt ale unor antiviruși și numai apoi infectează. Este și cazul virusului *WereWolf* care evită infectarea fișierelor: *CLEAN - McAfee CLEAN, AVP - Antiviral Toolkit Pro, TB - ThunderByte Antivirus, QB - Qbasic, SCAN - McAfee SCAN, NAV - Norton Antivirus, V-* orice fișier care începe cu 'V', *FINDV - S&S Findvirus, GUARD - S&S VirusGuard, FV - S&S Findvirus, CHKDS - Chkdsk*-ul *DOS, F-PR - F-Prot, -D- AVP TSR.* Se întâlnesc și viruși care nu permit lansarea unor programe antivirus afișând diferite mesaje de eroare. Primul virus *stealth* a fost virusul *Brain* apărut în ianuarie 1986 provenind din Pakistan și aproape toate variantele sale nu sunt dăunătoare.

Tunneling

Este o tehnică specială utilizată de viruși pentru a nu fi descoperiți de către antiviruși, mai ales cei de tip *shield.* Antivirușii rezidenți intercepteză permanent operațiile de accesare a fișierelor, verificând dacă acestea conțin viruși. Utilizând tehnica de *tunneling*, virușii se folosesc de o tratare hardware specială, suportată de familia de procesoare *x86*, pentru a afla adresa efectivă a diverselor servicii sistem *DOS* și *BIOS.* În acest mod virușii folosesc serviciile sistem fără ca antivirusul să observe acest lucru.

Autocriptare

Caracteristica principală a virușilor este cea de răspândire. Antivirușii detectează infecțiile prin căutarea unei amprente specifice fiecărui virus. Virușii infectează fișierele inserând propriul cod în fișierele infectate care este același în fiecare fișier virusat. Dar pentru a nu fi descoperiți, ei se autocriptează, iar unii își chiar modifică codul la infectare. Astfel, virusul nu va mai putea fi recunoscut după o anumită amprentă. Mulți viruși își criptează codul utilizând un algoritm simplu, cel mai simplu fiind cu operatorul XOR. Algoritmul de criptare poate avea și o cheie care de cele mai multe ori este ora și data ceasului intern. Acesta face ca un singur virus să poată avea mai multe forme, exceptând rutina de decriptare care este aceeași, care poate fi folosită de antiviruși pentru recunoaștere. Pentru a împiedica acest lucru, creatorii de viruși încearcă să minimizeze codul virusului. Aceasta se bazează pe faptul că amprentele nu pot conține un cod foarte scurt din virus, deoarece posibilitatea întâlnirii unor asemenea coduri este ridicată, antivirușii cauzând alarme false.

Polimorfism

Este o tehnică mai avansată decât autocriptarea. Față de autocriptare virușii polimorfici își modifică și rutina de decriptare. Astfel, un virus va avea un număr foarte mare de forme, cel puțin 255. Mai mult, aceștia pot adăuga aleator secvențe de cod, fără nici o importanță, de ordinul a câtorva bytes, numite *trash.* Prin utilizarea acestor secvențe de cod se modifică și dimensiunea virusului, același virus având mai multe dimensiuni. Antivirușii vor detecta foarte greu acest tip de viruși care sunt complet criptați și având dimensiuni variabile. Creatorul de viruși bulgar, ***Dark Avenger***, a creat în anul 1991 un motor de mutații numit *MtE. MtE* poate fi încorporat în orice virus, virusul devenind polimorfic. Motorul de mutații a fost distribuit în underground-ul *Internet*-ului ca un fișier obiect care era adăugat oricărui virus, nou sau vechi. Algoritmul de criptare, utilizat, de *MtE,* este unul foarte dezvoltat încât un singur byte comun întâlnindu-se în toate criptările *MtE.* Instrucțiunea comună este JNZ (*Jump if Not Zero*), care poate fi întâlnită oriunde este posibil în program. *MtE* folosește mulți și diferiți algoritmi de criptare și adaugă aleator doi bytes la rutina de decriptare. Dar *MtE* nu este unicul motor de criptare. Există foarte multe asemenea motoare mutagene dintre care pot fi amintite: *DSME - Dark Slayer Mutation Engine, TPE - TridenT Polymorphic Engine* scris în 1992 de ***Masud Khafir***, *DSCE - Dark Slayer Confusion Engine* care este o versiune rescrisă a lui *DSME, DAME* - Dark *Angel's Multiple Encryptor, MutaGen* creat de *MnemoniX.*

Efectele virușilor

Prezența unui virus poate fi descoperită în urma mai multor simptome:

- creșterea memoriei utilizate;
- scăderea vitezei de operare a calculatorului;
- timp suplimentar la execuția programelor;
- accesarea excesivă a harddiscului sau a unei dischete protejate în unitatea de disc;
- programele nu mai pot fi executate (cazul virușilor *overwrite*);
- erori anormale de protecție la scriere;
- imposibilitatea de pornire a *Windows*-ului;
- blocarea *Windows*-ului prin accesarea directă a hardware-ului de către viruși;
- avertismentul *Windows* care semnalează faptul că nu se utilizează accesul pe 32 de biți;
- imposibilitatea salvării documentelor decât ca *template.*

Pe plan internațional se observă o prezență tot mai rară a virușilor de *boot* și de program, datorată utilizării din ce în ce mai puțin a dischetelor și creșterii interconectivității calculatoarelor prin utilizarea tot mai accentuată a rețelelor, mai ales a *Internet*-ului. O dată cu trecerea la noile platforme *Windows*, în special *Windows '95* și *Windows '98*, virușii s-au specializat pe aceste platforme, locul virușilor *DOS* fiind luat de virușii ce infectează aplicațiile *Windows.* Pe măsură ce virușii de *boot* și de program sunt tot mai rar întâlniți, au apărut virușii de *macro*, virușii de *Windows* și *viermii* ce se propagă prin *Internet*, care vor domina noua lume virală.

Flaviu Cibu și Adrian Bugariu sunt elevi în clasa a XII-a la Liceul Teoretic "Octavian Goga" din Sibiu și pot fi contactați la adresele flaviu@zxmail.com, *respectiv* adrianbugariu@crosswinds.net.

Ghiocelul, mărțișorul și ALGORITMII

Sorin Groza

Încet, încet căldura de la gura sobei este înlocuită de căldura soarelui de primăvară. Acum când zăpada începe să se topească, ghioceii și mărțișoarele sunt un simbol al unui nou început. Am vrea ca și algoritmii prezentați să vă folosească în această nouă perioadă de studiu și să contribuie la îmbogățirea cunoștințelor în ale informaticii. Și nu în ultimul rând, rezolvându-le, vă dovediți dragostea pentru informatică.

Ghiocei

De îndată ce zăpada a început să se topească și primii ghiocei să mijească de sub plapuma albă, baieții clasei a V-a B au hotărât să pornească la cules de ghiocei. Ei sunt foarte hotărâți să ofere colegelor de clasă cât mai mulți ghiocei. După ce fiecare a oferit un anumit număr de ghiocei, fetele au vrut să afle care este cea mai lungă secvență de băieți cu proprietatea că fiecare a oferit un număr de ghiocei mai mic decât următorul. *Prompter*, informaticianul clasei a fost rugat să scrie programul care să rezolve problema. Scrieți și voi acest program și comparați-l cu cel scris de *Prompter*.

Exemplu

Intrare

```
n = 6
15, 7, 10, 15, 8, 12
```

Ieșire

```
7 10 15
```

Această problemă reprezintă de fapt găsirea "celei (celor) mai lungi secvențe de elemente consecutive ordonate strict crescător". Deoarece pot să existe mai multe astfel de secvențe, pentru început vom afla lungimea acestor secvențe în variabila `max`. Pentru aceasta vom incrementa contorul `f` cât timp `x[i]<x[i+1]`. Când nu mai este îndeplinită această condiție înseamnă că s-a terminat o astfel de secvență. Dacă lungimea ei este mai mare decât `max` atunci vom păstra în `max` lungimea ei. Printr-o nouă parcurgere a șirului, vom afișa fiecare secvență a cărei lungime este egală cu `max`. Astfel, vor fi afișate toate secvențele ce îndeplinesc condiția cerută.

Listing GHIOCEI.PAS

```
var n,i,j,max,k,f:Integer;
    x:array[1..100] of Integer;
Begin
  Write('Numar elevi = '); Readln(n);
  Write('Dati numarul de ghiocei ai ');
  Writeln('fiecarui elev...');
  for i:=1 to n do Read(x[i]);
  f:=1;
  for i:=1 to n do
  begin
    if x[i]<x[i+1]
    then Inc(f)
    else
    begin
      if f>max then max:=f;
      if f>1 then Inc(k);
      f:=1
    end
  end;
  if k=0 then
    Writeln('Nu exista secventa!')
  else
  begin
    f:=1;
    for i:=1 to n do
    begin
      if x[i]<x[i+1] then Inc(f)
      else
      begin
        if f=max
        then
```

```
          begin
            for j:=i-f+1 to i do
              Write(x[j],' ');
            Writeln
          end;
          f:=1
        end
      end
    end
End.
```

Mărțișoare

La talcioc e agitație mare deoarece odată cu venirea primăverii au apărut mărțișoarele. Anul acesta sunt la mare modă mărțișoarele care au gravate pe ele diverse cuvinte. Vânzătorul a aflat că un grup de mai mulți prieteni doresc să aibă mărțișoare ale căror cuvinte să fie anagrame între ele, adică să fie formate din aceleași litere dar dispuse în altă ordine (de exemplu, *roma - amor - ramo*).

Deoarece sunt mulți cumpărători și cu toții se grăbesc, vânzătorul dorește să împartă mărțișoarele în grupe, astfel încât fiecare cuvânt de pe mărțișor să reprezinte o anagramă a celorlalte cuvinte din grup. Un grup trebuie să conțină cel puțin două cuvinte. Pentru că are foarte multe mărțișoare și timpul este scurt, vânzătorul vă roagă să-i scrieți un program care face acest lucru - evident, în schimbul unui mărțișor.

Exemplu

Intrare

```
n=10
aiud, xxl, dor, duia, xlx, sos, udai,
lxx, bona, daiu
```

Ieșire

```
aiud duia udai daiu
xxl xlx lxx
```

După citirea celor n cuvinte (șirul x) se face o "dublură" a acestuia pentru a nu pierde cuvintele originale (șirul y), deoarece fiecare cuvânt se ordonează astfel încât să conțină caracterele ordonate alfabetic. Astfel pentru a verifica dacă două sau mai multe cuvinte sunt anagrame ale unui același cuvânt, acestea se ordonează alfabetic și se verifică dacă sunt egale între ele.

Se vor ordona alfabetic, în paralel, șirul ce conține cuvintele în care caracterele sunt ordonate alfabetic și șirul cuvintelor inițiale.

După această operație cuvintele egale între ele vor ocupa poziții consecutive. Printr-o parcurgere a șirului cuvintelor ordonate alfabetic (șirul x) se observă dacă există mai multe cuvinte consecutive egale.

Dacă sunt mai mult de două, se va face tipărirea cuvintelor de pe pozițiile corespunzătoare, dar din șirul cu cuvintele inițiale (șirul y). În caz că nu există astfel de grupuri se va afișa un mesaj.

Listing MARTISOR.PAS

```
var n,i,j,k:Integer;
    aux:Char; xua,a:string;
    x,y:array[1..100] of string[100];
    gata,termin,este:Boolean;
Begin
  Write('Numarul martisoarelor = ');
  Readln(n);
  Write('Introduceti cuvintele de pe ');
  Writeln('martisoare...');
  for i:=1 to n do Readln(x[i]);
  y:=x;
  repeat
    gata:=true;
    for i:=1 to n do
    begin
      for j:=1 to Length(x[i])-1 do
        if x[i][j]>x[i][j+1] then
        begin
          aux:=x[i][j];
          x[i][j]:=x[i][j+1];
          x[i][j+1]:=aux; gata:=false
        end
    end
  until gata;
  repeat
    termin:=true;
    for i:=1 to n-1 do
      if x[i]>x[i+1] then
      begin
        xua:=x[i]; x[i]:=x[i+1];
        x[i+1]:=xua;
        a:=y[i]; y[i]:=y[i+1];
        y[i+1]:=a;
        termin:=false;
      end;
  until termin;
  k:=1; este:=false;
  for i:=1 to n do
  begin
    if x[i]=x[i+1] then inc(k)
    else
      begin
        if k>1 then
        begin
          for j:=i-k+1 to i do
            if i-(i-k)>1 then
              Write(y[j],' ');
          k:=1; este:=true; Writeln
        end
      end
  end;
  if not este then
    Writeln('Nu se pot face grupe.')
End.
```

La discoteca

La marea discotecă a primăverii și-au anunțat participarea foarte multe persoane. Organizatorii împreună cu D.J.-ul doresc ca totul să fie perfect. În vederea realizării unei petreceri de neuitat D.J.-ul dorește să cunoască unele detalii despre melodiile pe care le va prezenta. Astfel, el confruntă topul muzical de săptămâna aceasta cu topul muzical de săptămâna trecută. Neavând foarte mult timp la dispoziție, el îi cere unui programator (în schimbul intrării gratuite la discotecă) să scrie un program care, plecând de la cele două topuri, să stabilească următoarele:

- care sunt noile formații intrate în top;
- care sunt formațiile care au staționat, adică ocupă exact același loc ca și săptămâna trecută;
- care formație a urcat cele mai multe poziții în top și care este acest număr de poziții.

Exemplu

Intrare

```
n=6
abba, boyz, hanson, colio, sandra, queen
(topul săptămânii trecute)
sandra, boyz, queen, colio, hanson,
vanghelis (topul acestei săptămâni)
```

Ieșire

```
noi intrari: vanghelis
stationari: boyz colio
urcare maxima: sandra cu 4 pozitii
```

Se citesc componentele topurilor celor două săptămâni în șirurile x și y. Pentru a găsi noile intrări în top se caută fiecare element al șirului y în șirul x. Dacă nu există, înseamnă că este o nouă intrare și se va afișa pe ecran. Dacă elementele de pe poziții identice din cele două șiruri sunt egale atunci aceasta reprezintă o staționare, situație în care se va afișa această valoare. În ceea ce privește determinarea formației cu cea mai mare urcare în top, vom calcula elementul cu diferența cea mai mare dintre poziții în cele două șiruri. În variabila poz vom avea în final poziția în șirul y a formației cu cea mai mare urcare, iar în max vom avea această urcare.

Listing DISCOTECA.PAS

```
Program Discoteca;
var n,i,j,max,f,k,a,poz:Integer;
    x,y:array[1..100] of string;
    este:Boolean;
Begin
  Write('Numarul melodiilor =');
  Readln(n);
  Writeln('Topul saptamanii trecute:');
  for i:=1 to n do Readln(x[i]);
  Writeln('Topul acestei saptamani:');
  for i:=1 to n do Readln(y[i]);
        { determinarea noilor intrari }
  for i:=1 to n do
    if x[i]<>y[i]
      then Inc(k);
  if k=0 then
    Write('Nu au fost noi intrari');
  if k>0 then
  begin
    Write('Noi intrari: ');
    for i:=1 to n do
    begin
      este:=true;
      for j:=1 to n do
        if x[j]=y[i] then
          este:=false;
      if este then Write(y[i],' ')
    end
  end;
    { sfarsit afisarea noilor intrari }
  Writeln;
        { determinarea stationarilor }
  for i:=1 to n do
    if x[i]=y[i] then Inc(f);
  if f=0 then
    Write('Nu sunt stationari')
  else
    begin
      Write('Stationari: ');
      for i:=1 to n do
        if x[i]=y[i] then
          Write(x[i],' ')
    end;
    { sfarsit afisarea stationarilor }
  Writeln;
        { determinarea urcarii maxime }
  for i:=1 to n do
  begin
    for j:=1 to n do
      if x[j]=y[i] then a:=j-i;
    if a>max then
      begin max:=a; poz:=i end
  end;
  Write('urcare maxima: ',y[poz],
                  ' cu ',max,' locuri')
      { sfarsit afisare urcare maxima }
End.
```

Pentru consolidarea cunoștințelor în informatică și pentru pregătirea așa cum se cuvine a concursurilor care vă așteaptă, vă propunem spre rezolvare următoarele probleme:

Despărțirea în silabe

Prichindel este foarte bucuros deoarece a învățat despărțirea cuvintelor în silabe. Pentru a se verifica asupra temelor pe care le are de făcut, el îl roagă pe fratele lui mai mare să-i

facă următorul program: se introduc *n* cuvinte despărțite în silabe, ce formează un dicționar și un text ce conține mai multe fraze. *Prichindel* va descompune în silabe cuvintele acestui text și va face confruntarea cu soluția dată de program. Acesta va descompune cuvintele conform dicționarului. Dacă un cuvânt nu apare în dicționar se va da mesajul `Dictionar incomplet`.

Exemplu

Intrare

```
n=7
ma-ma, din, fa-vo-rit, ca-ba-na, mun-te,
eu, pe-ru-ca - dicționarul
cabana din munte. - textul
```

Ieșire

```
ca-ba-na din mun-te.
```

Intervale

La matematică, elevii au învățat despre intervale. Fiecare elev a trebuit să indice câte un interval de forma `[x,y]`. Ei au învățat la școală că un interval poate fi inclus într-un alt interval (de exemplu, intervalul `[5,8]` este inclus în intervalul `[2,10]`). Scrieți un program care citește cele `n` intervale și determină acel interval care este inclus în cele mai multe intervale. Programul va afișa acest interval și numărul de intervale în care este inclus.

Exemplu

Intrare

```
n=5
[8,40], [10,20], [1,99], [6,25], [15,20]
```

Ieșire

```
[10,20] - 3
```

Excursia

Venirea primăverii coincide și cu plecarea în excursii. Pentru a pleca într-o excursie, la cules de ghiocei, elevii unei clase și-au scos alocația, fiecare având o sumă de bani . Scrieți un program care determină cel mai mare grup de copii ale căror sume de bani se divid fiecare cu un același număr. Se vor afișa pe ecran sumele de bani ale copiilor care formează acest grup.

Exemplu

Intrare

```
n=9
7, 15, 20, 6, 105, 18, 40, 111, 35
```

Ieșire

```
15 20 105 40 35
```

Domnul Sorin Groza este profesor la Clubul Copiilor din Aiud și poate fi contactat prin intermediul redacției.

news...

AMD K6-2 550

AMD a preluat din nou conducerea și în războiul procesoarelor ieftine, lansând pe data de 22 februarie 2000 un procesor din familia K6-2 care funcționează la o frecvență de aproximativ 550 MHz.

Potrivit oficialilor companiei acest nou procesor este perfect compatibil cu sistemul de operare Windows 2000, dar și cu Windows '98 și alte platforme.

Procesorul K6-2 550 oferă suport 3DNow!, conține 9.3 milioane de tranzistori și este fabricat pe baza tehnologiei de 0.25 microni.

La data lansării un procesor AMD K6-2 550 avea prețul de 189$ (pentru cantități de minimum 1000 de bucăți).

La ora actuală compania rivală, Intel, oferă procesoare Celeron cu frecvențe de până la 533 Mhz, dar a promis că va lansa în curând pe piață produse mai rapide din această familie.

Intel va trece la tehnologia de 0.18 microni și pentru procesoarele Celeron, fapt care va permite mărirea vitezei acestora. Compania a anunțat că în curând vor fi disponibile procesoare Celeron cu frecvențe de 600 și 667 MHz, dar nu a fost anunțată o dată exactă a lansării acestora.

Totuși, trecerea la această nouă tehnologie va mări considerabil costurile de producție, ceea ce va determina o creștere a prețurilor.

COMPUTER PRESS AGORA

PC REPORT
ECHIPAMENTE · SISTEME · APLICAȚII · REȚELE · INTERNET
Vol. 9, Nr. 1. Ianuarie 2000 LEI 20.000

Din acest număr:

- **Unelte de modelare UML**
- **Introducere în Python**
- Ghid anti-hacker
- Solid Edge 7
- Standarde GNU
- XML prin SAX
- PC Concrete

www.pcreport.ro

Revista
specialiștilor
în tehnologia informației și comunicațiilor

nologia informației și comunicat

Cure-all?

For a great night's sleep, try Compaq Deskpro. It relieves the headaches of day-to-day PC management with powerful features like the new Compaq Insight Manager LC. It's the industry's first integrated tool for managing workgroups of PCs from a simple, browser-based interface. So you spend less time playing doctor and more time getting real work done. Just another healthy dose of exceptional value from Compaq - and another reason Deskpro is number one. For more therapeutic thoughts, visit www.compaq.com/deskpro, or contact us at phone: +40-1-3033100

intel inside
pentium III

COMPAQ
prin distribuitorul autorizat

OMNILOGIC BGS
135, Bucureşti-Ploieşti Road,
Bucharest 1, Romania
PO BOX 18-131
Tel: +40-1-303 31 00
Fax: +40-1-303 31 64
www.omnilogic.ro

Intel, the Intel Inside logo and Pentium are registered trademarks of Intel Corporation

Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

Aprilie 2000
Vol. 10/4

Lei 15.000

din sumarul numărului:

TEHNO

Instrucțiunea SELECT SQL

SOLUȚII

Soluțiile problemelor propuse CEOI '99

SERIAL

Pagini Web cu XML 1.0

Introducere în teoria grafurilor

Internet

Concepție, arhitectură și funcționare

Bac: Bacalaureat 2000

ACTUAL

Planuri pentru anul 2000

PC MAGAZINE ROMÂNIA

COMPUTER PRESS AGORA

Din acest număr:

- **Windows 2000**
- **Chat-ul productiv**
- **Plăci grafice**
- **Sănătatea și Web-ul**

www.pcmagazine.ro

Revista utilizatorilor de calculatoare personale

COMPUTER PRESS AGORA

Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

Vol. 10, Nr. 4/2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Cristian Cadar (București)
Ion Cozac (Târgu Mureș)
Marin Vlada (București)
Bogdan Dumitru (Ploiești)

Tehnoredactare:
Mihai Uricaru, Călin Borz
Corectura:
Carmen Bucur
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Gabriela Bucșa (gbucsa@agora.ro)
Gabriel Piscupescu (gpiscup@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Anamaria Pascu (apascu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Mihaela Szabo (mszabo@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

editorial.. PAG. 3

CERF 2000

Cea mai importantă expoziție IT&C din România

soluții.. PAG. 5

Probleme REZOLVATE

Olimpiada de Informatică a Europei Centrale

Bogdan DUMITRU

concurs.. PAG. 17

Rezultatele rundei a cincea

Clasament

Soluțiile de la runda a cincea

Descalificări

bac.. PAG. 22

BACALAUREAT 2000

Al doilea bacalaureat în informatică

tehno.. PAG. 29

Instrucțiunea SELECT SQL

Fundamentele bazelor de date

Ion COZAC

serial.. PAG. 34

Introducere în teoria GRAFURILOR
Cristian CADAR

Pagini WEB cu XML 1.0
Mihai SCORȚARU

actual.. PAG. 43

Planuri pentru anul 2000

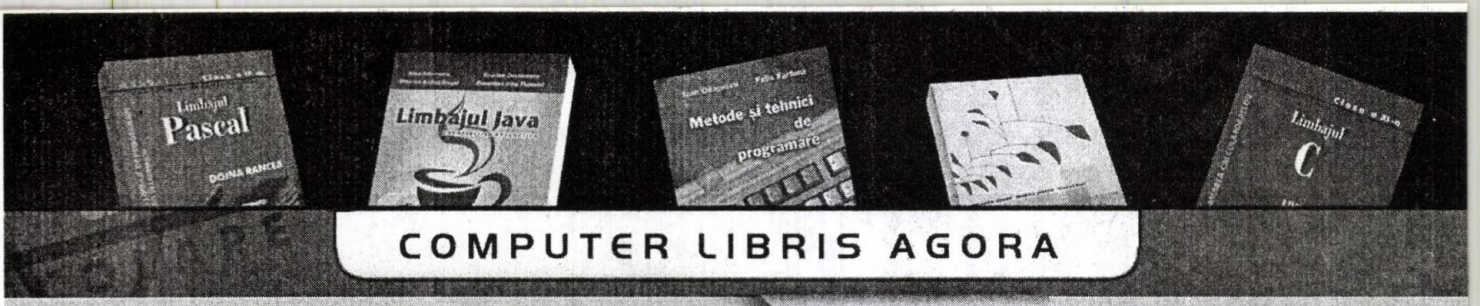
Procesoare de la Intel și AMD

Mihai SCORȚARU

internet.. PAG. 45

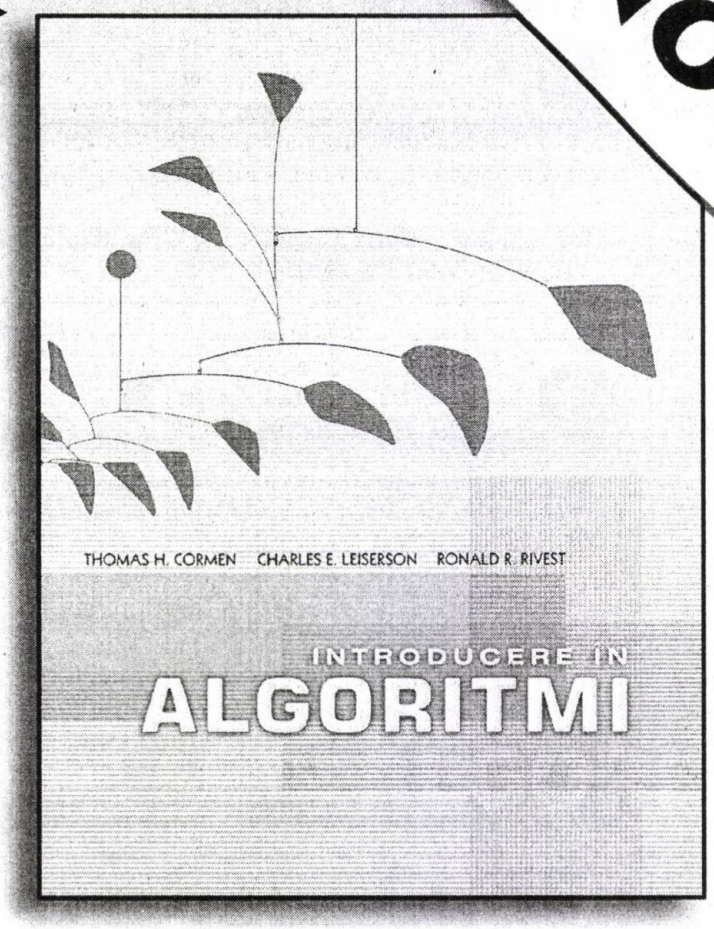
CONCEPȚIE, ARHITECTURĂ și FUNCȚIONARE

Marin VLADA



COMPUTER LIBRIS AGORA

NOUTATE



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paralleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

CERF 2000

În perioada 11-15 aprilie 2000 a avut loc la București a noua ediție a *Expoziției și Conferinței Internaționale Specializate de Tehnologia Informației și Comunicații*, CERF (*Computer and Electronics Romanian Fair*).

CERF 2000 a avut loc la complexul expozițional *RomExpo*, domeniile prezentate la această ediție fiind tehnologia informației, birotică, mass media de specialitate. Tematica CERF a fost comunicații de date, telecomunicații, rețele de calculatoare, software, hardware, periferice, multimedia, consumabile, servicii în domeniu. Secțiunile specializate din acest an au fost *echipamente și servicii în domeniul comunicațiilor* și *industria românească de software și hardware*. La această ediție au participat aproximativ 350 de firme expozante, iar numărul vizitatorilor este estimat la 80.000.

De la această expoziție nu putea lipsi ***Computer PRESS Agora*** care a acordat din nou distincțiile ***Premiile de tehnologie PC Report la CERF 2000*** și a realizat un program de conferințe foarte interesant.

Cu această ocazie a avut loc lansarea oficială a cărții ***Introducere în Algoritmi*** scrisă de ***Thomas H. Cormen, Charles E. Leiserson*** și ***Ronald R. Rivest***, o carte de referință în domeniul algoritmicii, tradusă pentru prima dată în limba română. Cartea s-a bucurat de un succes care a depășit așteptările, numărul celor interesați de acest subiect fiind surprinzător de mare. Chiar dacă nu oricine își poate permite să cumpere o astfel de cărte am fost plăcut surprins când am văzut că există chiar și elevi care au făcut un efort financiar considerabil pentru a achiziționa cartea.

De un succes deosebit s-a bucurat și cartea ***Limbajul Java, o abordare sistemică*** scrisă de un colectiv condus de ***prof. Irina Athanasiu*** de la *Universitarea Politehnică* din București, carte care se află deja la cea de-a doua ediție. Numărul celor care doresc să cunoască acest limbaj de programare este foarte mare, iar prețul cărții este accesibil pentru toți cei interesați.

Tot la CERF a fost lansată și cartea ***Un atelier de programare*** scrisă de ***prof. Cristian Giumale*** de la *Universitatea Politehnică* din București. Cartea este o culegere de studii de caz cunstruite pe baza problemelor propuse la concursurile *Southeastern European Regional Programming Contest*, din perioada 1995-1999, organizate sub egida ***ACM***.

Sursele programelor care apar în carte sunt disponibile pentru download la adresa http://libris.agora.ro. Această carte a reprezentat un punct de interes pentru cei care se pregătesc pentru concursurile de informatică care urmează. Numărul acestora mă face să cred că România se va afla din nou pe podiumurile de premiere la olimpiadele internaționale din următorii ani.

În rest ediția din acest an nu s-a deosebit prea mult de edițiile anterioare; a fost lume multă, noutăți puține și expozanți care îți ofereau tot felul de materiale promoționale deosebit de "interesante".

Pe data de 13 aprilie expoziția a fost vizitată și de președintele României, domnul ***Emil Constantinescu*** care s-a arătat foarte interesat de domeniul *IT&C*. Ar fi bine dacă acest interes s-ar concretiza printr-o implicare reală în dezvoltarea sistemului informațional de la noi.

La CERF au fost prezente și companiile românești de telefonie mobilă care și-au prezentat ofertele la fel ca de fiecare dată. Totuși a ieșit în evidență *Cosmorom* care încerca să ne convingă că serviciile sale sunt mai bune decât cele oferite de concurență. Acum că avem patru furnizori de astfel de servicii s-ar putea ca tot mai mulți utilizatori să beneficieze de ele. Totuși, *Suntel* și *Cosmorom* nu se pot ridica încă la nivelul la care se află *Connex* și *Dialog*, dar faptul că există alternative nu poate decât să ne bucure. Nu a lipsit nici *RomTeleCom*, dar acest unic furnizor de servicii de telefonie fixă de la noi nu a anunțat nimic nou. Deci, prețul pentru conexiunile *Dial-up* rămâne același și cei interesați își vor pierde în continuare nopțile pentru a utiliza *Internet*-ul.

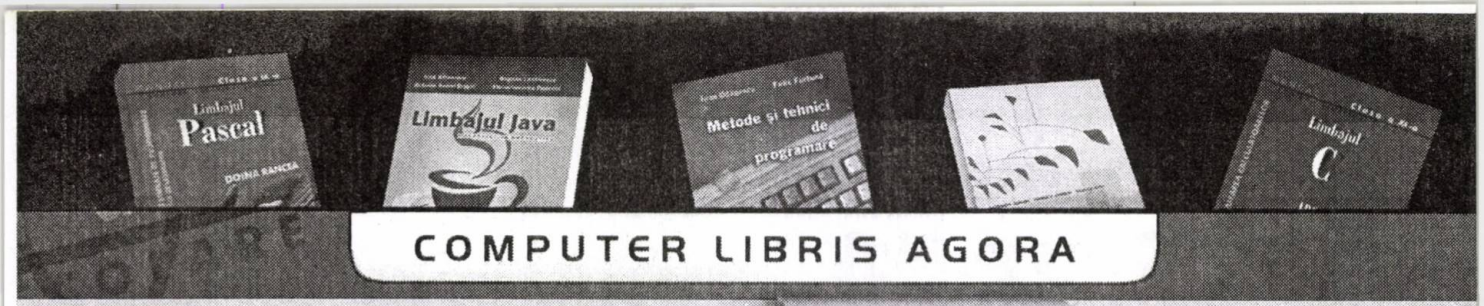
În ceea ce privește sistemele de operare, *Microsoft Windows 2000* nu a avut nici un contracandidat serios, *Linux*-ul lipsind aproape cu desăvârșire de la CERF.

Ofertele de procesoare nu au fost nici ele spectaculoase. Chiar dacă nu ne așteptam să putem vedea aici un procesor de 1 GHz la lucru, nici nu puteam să credem că cel mai bun procesor prezentat la CERF a fost un *AMD Athlon 800* și nu exista nici un procesor *Intel* care să fi fost lansat în anul 2000.

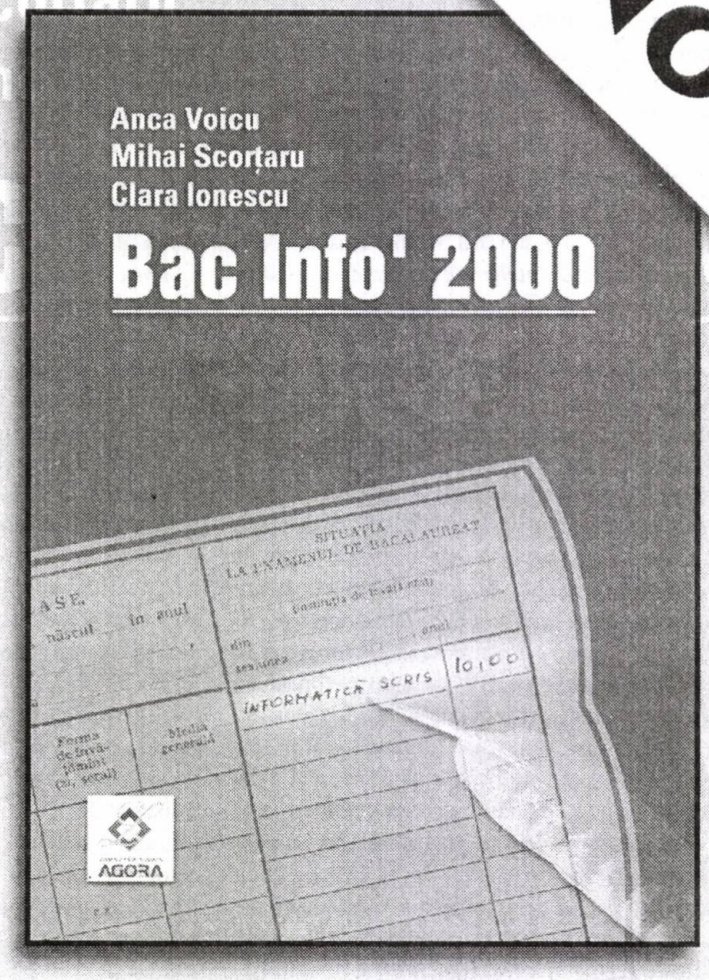
Totuși CERF este cel mai important târg de acest gen de la noi din țară și ar trebui să fim bucuroși că există. Ar fi mult mai bine dacă am avea și noi mai multe astfel de expoziții deoarece acestea reprezintă o ocazie extraordinară de a atrage atenția asupra domeniului IT.

Se pare că majoritatea firmelor participante au fost mulțumite de rezultatele obținute la CERF. Nu ne rămâne decât să așteptăm și să vedem ce se va întâmpla în toamnă la IFABO.

Mihai Scorțaru



NOUTATE



În această cărticică veți găsi subiecte de Bac la informatică și nu doar atât.
În partea a doua, vă punem la dispoziție posibile răspunsuri, soluții ale problemelor. Deoarece limbajul de programare este la alegere, depinde de candidat pe care îl preferă. Acolo unde răspunsul depinde de limbajul ales, veți găsi soluțiile corespunzătoare prezentate și în Pascal și în C.
La sfârșitul cărții sunt patru "variante" de subiecte propuse ale căror rezolvări le veți putea citi și studia în Gazeta de Informatică.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

SOLUȚIILE problemelor propuse

Vă prezentăm în continuare soluțiile problemelor de la Olimpiada de Informatică a Europei Centrale din 1999. Rezolvările aparțin lui Bogdan Dumitru din Ploiești, medaliat cu argint la CEOI '99

P010001: Livrări

Această problemă, deși a avut un text greoi, a fost cea mai simplă (și evident cea mai slab punctată) dintre cele 6 probleme prezentate concurenților spre rezolvare. Specificația *toate posibilitățile* indică clar metoda de rezolvare: *backtracking*. Tot ce mai trebuia făcut era optimizarea programului pe cât posibil, având în vedere numărul mare de soluții posibile.

Procedura `Load` citește caracterele puse la dispoziție, memorând în vectorul `NR` câte "exemplare" avem din fiecare caracter. Procedura `Preprocess` construiește vectorii `CAR` și `CATE`; în `CAR` reținem caracterele distincte sortate crescător (numărul lor este reținut de variabila `L`); în `CATE` reținem de câte ori avem fiecare caracter. Numărul total de caractere este `S`.

Procedura recursivă `Rec` implementează metoda *backtracking* propriu-zisă. Dacă am ajuns pe nivelul `S+1`, am obținut o soluție, pe care o tipărim; dacă nu, completăm nivelul curent (notat `K`) cu toate caracterele avute la dispoziție în momentul respectiv, în ordinea crescătoare indusă de vectorul `CAR` și reapelăm procedura, nu înainte de a decrementa corespondentul din `CATE` al caracterului ales. Ordinea crescătoare de alegere a caracterelor garantează ordinea lexicografică a soluțiilor.

Procedura `Bkt` deschide fișierul de ieșire la scriere și îi asociază un buffer (necesar având în vedere dimensiunea fișierului - maxim `2MB`) și lansează procedura recursivă. Fiecare soluție este scrisă de procedura `Scrie`, apelată ori de câte ori se validează condiția de ieșire din recursivitate. Complexitatea soluției: pentru fiecare caracter adăugat unei soluții se testează cel mult toate caracterele distincte, care sunt în număr de cel mult 26 (constanta notată `NMAX`); numărul de operații efectuate devine astfel proporțional cu *O (Output * NMAX)*, unde `Output` semnifică dimensiunea fișierului de ieșire. Această complexitate (minimă de altfel) asigură trecerea cu brio a tuturor testelor propuse în concurs.

Listing LIVRARI.PAS

```
program orders;
const ni='ORDERS.IN';
      no='ORDERS.OUT';
      nmax=26;
      tmax=200;
var f:Text;
    buffer:array [1..60000] of Char;
    nr:array ['a'..'z'] of Integer;
    car:array [1..nmax] of Char;
    cate:array [1..nmax] of Integer;
    out:array [1..tmax] of Char;
    ch:Char;
    i,l,s:Integer;

procedure load;
begin
  for ch:='a' to 'z' do
    nr[ch]:=0;
  Assign(f,ni); Reset(f);
  while not Seekeoln(f) do
  begin
    Read(f,ch);
    nr[ch]:=nr[ch]+1;
  end;
  Close(f)
end;

procedure preprocess;
begin
  l:=0;
```

```
  s:=0;
  for ch:='a' to 'z' do
    if nr[ch]>0 then
    begin
        l:=l+1;
        car[l]:=ch;
        cate[l]:=nr[ch];
        s:=s+cate[l]
    end
end;

procedure scrie;
begin
  for i:=1 to s do
    Write(f,out[i]);
  Writeln(f)
end;

procedure rec(k:Integer);
var i:Integer;
begin
  if k>s then scrie
  else
    for i:=1 to l do
      if cate[i]>0 then
      begin
        out[k]:=car[i];
        cate[i]:=cate[i]-1;
        rec(k+1);
        cate[i]:=cate[i]+1
      end
end;

procedure bkt;
begin
  Assign(f,no);
  Settextbuf(f,buffer);
  Rewrite(f);
  rec(1);
  Close(f)
end;

Begin
  load;
  preprocess;
  bkt
End.
```

P010002: Numere de telefon

Aceată problemă a urmărit să testeze cunoștințele de programare dinamică ale concurenților, precum și abilitatea de a manipula structuri mari de date (de ordinul sutelor de KB).

Cele maxim 50.000 de cuvinte de maxim 50 de caractere fiecare nu puteau fi memorate într-un vector de pointeri la string-uri, din simplul motiv că ar fi fost necesari 50.000*50=2.500.000 octeți, adică peste 2.400 KB de memorie. Nici listele nu reprezentau o soluție, o "legătură" între 2 celule ocupând 4 octeți.

Singura soluție de memorare era o matrice de caractere, în care cuvintele erau scrise "la rând". Soluția se bazează pe faptul că lungimea totală a cuvintelor nu depășea 300.000. În plus, trebuia reținut un vector de 50.000 de elemente, care să conțină lungimile cuvintelor. Aparent, metoda prezintă multe dezavantaje, ca de exemplu faptul că accesarea unui cuvânt oarecare nu se poate face decât după ce se calculează *deplasamentul* lui față de prima celulă din matrice, nemaivorbind de faptul că respectivul cuvânt s-ar putea să ocupe mai multe linii din matrice. De fapt, algoritmul descris mai jos va folosi eficient această matrice.

Metoda de rezolvare este următoarea: pentru fiecare poziție din numărul de telefon reținem numărul minim de cuvinte cu care putem să codificăm numărul până la respectiva poziție, precum și ultimul cuvânt folosit (pentru reconstituire). Baleiem numărul de telefon și, din poziția curentă, încercăm să optimizăm, folosind toate cuvintele avute, obținute baleind matricea creată. Algoritmul este o variantă de programare dinamică, cunoscută sub numele de *metoda înapoi*. Dacă reușim să ajungem la ultima poziție, există soluție; în caz contrar, nu.

Procedura Load citește cuvintele date și le memorează în matrice. Am ales ca matricea să aibă 30 de linii a câte 10.000 de caractere fiecare. Astfel, un cuvânt (a cărui lungime maximă este 50) se va afla pe cel mult două linii ale matricei.

Procedura Init_OK inițializează matricea OK, utilizată pentru a testa dacă este posibilă sau nu codificarea unei anumite cifre printr-un anumit caracter.

Procedura Init initializeaza vectorul V (în V[i] reținem numărul minim de cuvinte folosite pentru a codifica subșirul format cu primele i cifre din numărul de telefon).

Procedura Work încearcă, având soluția optimă pentru primele i cifre, să completeze soluția cu toate cuvintele avute. Acestea sunt considerate pe rând (citind matricea) și testate cu ajutorul matricei OK. Dacă un anumit cuvânt se potrivește, se face actualizarea.

Procedura Imposibil determină dacă problema nu are soluție, caz în care scrie un mesaj corespunzător și suspendă execuția programului.

Procedura Reconst reconstituie soluția, procedeu deloc banal. Mai întâi se determină numerele de ordine ale cuvintelor folosite (în vectorul USE), apoi acestea se sortează (în vectorul CUSE).

În procedura Scrie se baleiază matricea de cuvinte; când se găsește un cuvânt care a fost folosit în obținerea soluției, acesta este scris în fișierul de ieșire.

Pentru fiecare din cele maxim 100 de cifre ale numărului de telefon, se baleiază matricea de cuvinte, a cărei dimensiune poate fi cel mult 300.000. Complexitatea poate fi astfel *O(Numărul de Cifre * Input)*. În practică, optimizările făcute (se iese din ciclu la prima nepotrivire găsită)

coboară mult timpul de execuție. Sursa rulează aproape instantaneu pe întreg setul de teste propuse în concurs.

Listing TELEFON.PAS

```
Program phone;
const  ni='PHONE.IN';
       no='PHONE.OUT';
       pmax=100;
       nmax=50000;
       cmax=10000;
       lmax=30;

type sir=array [1..cmax] of char;
     psir=^sir;
     mat=array [1..lmax] of psir;
     vector=array [1..nmax] of byte;
     pvector=^vector;

var f:Text;
    a:mat;
    lng:pvector;
    ok:array [0..9,'a'..'z'] of
                              Longint;
    nr,use,cuse,stl,stc:
          array [1..pmax] of Longint;
    v,used,jump:array [0..pmax] of
                              Longint;
    numar,vali:string;
    n,l,c,i,j,k,h,p,cate,aux,sl,sc:
                              Longint;
    er:Integer;
    ch:Char;
    test:Boolean;

procedure load;
begin
  Assign(f,ni);
  Reset(f);
  Readln(f,numar);
  Readln(f,n);
  New(lng);
  l:=1;
  c:=0;
  New(a[1]);
  for i:=1 to n do
  begin
    readln(f,vali);
    lng^[i]:=Length(vali);
    for j:=1 to Length(vali) do
    begin
      c:=c+1;
      if c>cmax then
      begin
        l:=l+1;
        c:=1;
        New(a[l])
      end;
      a[l]^[c]:=vali[j]
    end
  end;
  Close(f)
end;

procedure init_ok;
begin
  for i:=0 to 9 do
    for ch:='a' to 'z' do
      ok[i,ch]:=0;
  ok[1,'i']:=1; ok[1,'j']:=1;
  ok[2,'a']:=1; ok[2,'b']:=1;
  ok[2,'c']:=1; ok[3,'d']:=1;
  ok[3,'e']:=1; ok[3,'f']:=1;
  ok[4,'g']:=1; ok[4,'h']:=1;
  ok[5,'k']:=1; ok[5,'l']:=1;
  ok[6,'m']:=1; ok[6,'n']:=1;
  ok[7,'p']:=1; ok[7,'r']:=1;
  ok[7,'s']:=1; ok[8,'t']:=1;
  ok[8,'u']:=1; ok[8,'v']:=1;
  ok[9,'w']:=1; ok[9,'x']:=1;
  ok[9,'y']:=1; ok[0,'o']:=1;
  ok[0,'q']:=1; ok[0,'z']:=1
end;

procedure init;
begin
  p:=Length(numar);
  for i:=1 to p do
    Val(numar[i],nr[i],er);
  v[0]:=0;
  for i:=1 to p do
    v[i]:=Maxint
end;

procedure work;
begin
  for i:=0 to p-1 do
    if v[i]<>maxint then
    begin
      l:=1; c:=0;
      for j:=1 to n do
      begin
        sl:=1;
        sc:=c;
        if i+lng^[j]<=p then
          if v[i]+1<v[i+lng^[j]] then
          begin
            test:=true;
            for k:=1 to lng^[j] do
            begin
              c:=c+1;
```

```
                if c>cmax then
                begin
                  l:=l+1;
                  c:=1;
                end;
                if ok[nr[i+k],
                      a[l]^[c]]=0 then
                begin
                  test:=false;
                  k:=lng^[j]
                end;
              end;
              if test then
              begin
                v[i+lng^[j]]:=v[i]+1;
                used[i+lng^[j]]:=j;
                jump[i+lng^[j]]:=i
              end
            end;
            l:=sl; c:=sc+lng^[j];
            if c>cmax then
            begin
              l:=l+1;
              c:=c-cmax
            end
        end
    end
end;

procedure imposibil;
begin
  if v[p]=maxint then
  begin
    Assign(f,no); Rewrite(f);
    Writeln(f,'No solution.');
    Close(f); Halt
  end
end;

procedure reconst;
begin
  i:=p; cate:=0;
  while i>0 do
  begin
    cate:=cate+1;
    use[cate]:=used[i];
    i:=jump[i];
  end;
  for i:=1 to cate do
    cuse[i]:=use[i];
  for i:=1 to cate-1 do
    for j:=i+1 to cate do
      if cuse[i]>cuse[j] then
      begin
        aux:=cuse[i];
        cuse[i]:=cuse[j];
        cuse[j]:=aux
      end;
  l:=1;
  c:=0;
  k:=1;
  for i:=1 to n do
    if k>cate then i:=n
    else
    begin
      if i=cuse[k] then
      begin
        stl[k]:=l;
        stc[k]:=c;
        k:=k+1
      end;
      c:=c+lng^[i];
      if c>cmax then
      begin
        l:=l+1;
        c:=c-cmax
      end
    end;
  for i:=1 to cate do
    if cuse[i]=cuse[i-1] then
    begin
      stl[i]:=stl[i-1];
      stc[i]:=stc[i-1]
    end
end;

procedure scrie;
begin
  Assign(f,no); Rewrite(f);
  for i:=cate downto 1 do
  begin
    for j:=1 to cate do
      if use[i]=cuse[j] then k:=j;
    l:=stl[k];
    c:=stc[k];
    for h:=1 to lng^[use[i]] do
    begin
      c:=c+1;
      if c>cmax then
      begin
        l:=l+1;
        c:=1
      end;
      Write(f,a[l]^[c])
    end;
    Write(f,' ')
  end;
  Writeln(f);
  Close(f)
end;
```

```
Begin
  load;
  init_ok;
  init;
  work;
  imposibil;
  reconst;
  scrie
End.
```

010003: Joc de paritate

Această problemă a fost considerată de majoritatea concurenților drept cea mai "inteligentă" problemă dintre cele propuse spre rezolvare în prima zi de concurs.

După lectura textului se conturează o primă idee de rezolvare: trebuie să concatenăm intervalele date pentru care limita superioară a unuia și limita inferioară a celuilalt sunt numere consecutive, în încercarea de a obține o contradicție (un interval despre care să știm că include atât un număr par, cât și un număr impar de numere 1 - lucru evident imposibil).

Această idee prezintă un dezavantaj imens: ea creează noi și noi intervale, existând multe situații pentru care memoria pusă de obicei la dispoziție (câteva sute de KB) nu este deloc suficientă.

Remarcăm că în loc de a concatena intervale, obținând intervale noi, tot mai mari, ar fi mult mai bine dacă am "scădea" intervale, obținând intervale mai mici, cu care se lucrează mai ușor. Vom spune, pentru simplificarea exprimării, că un interval este *par*, respectiv *impar*, după cum el conține un număr par, respectiv impar de cifre 1. Principiul de bază al rezolvării va fi: *având parităţile intervalelor* `[X,Y]` *şi* `[X,Z]`, *cu* `Y<Z`, *reţinem parităţile intervalelor* `[X,Y]` *şi* `[Y+1,Z]` *(adică a intervalului de lungime minimă şi a "diferenţei" între cele două intervale)*. Evident, orice prelucrare de intervale nu se poate baza decât pe coincidențe ale limitelor intervalelor (în afara acestora neputând apărea nici o contradicție).

Mai rămâne de văzut cum "reducem" într-o manieră eficientă intervalele.

Soluția cea mai bună pare a fi următoarea: reținem un vector A în care fiecare element reprezintă un interval (dat prin limite și paritate). Vectorul este permanent sortat după limita de început a intervalelor. La citirea unui nou interval, acesta este adăugat la sfârșitul vectorului.

Găsim apoi (dacă există) un interval care să înceapă în același punct cu intervalul citit. Notând cu I acest interval iar cu J noul interval, deosebim 3 cazuri:

1) limita superioară a intervalului I este mai mare decât limita superioară a intervalului J; în acest caz se inversează intervalele, urmând ca algoritmul să trateze cazul 3);
2) limita superioară a intervalului I este egală cu limita superioară a intervalului J; apar aici două subcazuri:
   2.1) dacă cele două intervale au aceeași paritate, noua informație este redundantă și este eliminată;
   2.2) în caz contrar, am detectat o contradicție; se afișează numărul de răspunsuri citite (1) și se suspendă execuția programului;
3) limita superioară a intervalului I este mai mică decât limita superioară a intervalului J; atunci intervalul J este înlocuit cu un nou interval, cu proprietățile:
   - limita inferioară este egală cu limita superioară a intervalului I, la care se adaugă 1;
   - limita superioară este egală cu limita superioară a intervalului J;
   - intervalul este *par* dacă intervalele I și J aveau aceeași paritate și *impar* în caz contrar;

Se continuă în același mod baleierea vectorului de intervale, reducând acolo unde este posibil.

Dacă se termină citirea, înseamnă că nu a fost detectată nici o contradicție. Se afișează deci numărul răspunsurilor citite și se termină execuția.

Procedura Load încarcă câte un interval care va fi prelucrat de procedura Work, cea mai importantă din program.

În această procedură (Work), noul interval este adăugat în poziția L a vectorului, nu înainte de a incrementa corespunzător variabila. Se parcurge mergând "înapoi" vectorul A (al intervalelor) până la găsirea primului interval pentru care limita inferioară este cel mult egală cu limita inferioară a ultimului interval citit. În funcție de relația între limita superioară a acestui interval și limita superioară a ultimului interval, apar trei cazuri, care se tratează așa cum a fost descris mai sus. Se trece apoi la următorul interval, operându-se în continuare toate reducerile posibile.

La detectarea unei contradicții, se apelează procedura Stop care scrie numărul răspunsurilor citite până atunci și întrerupere execuția programului.

Complexitatea maximă a acestui algoritm este $O(N^2)$. Însă, pe un anumit test, este foarte puțin probabil ca baleierea vectorului să se facă integral de fiecare dată (un astfel de caz ar fi acela în care intervalele sunt date în ordinea descrescătoare a limitelor inferioare). În orice caz, sursa prezentată rezolvă setul de teste propus în concurs folosind doar o parte din timpul de rulare maxim admis.

Listing PARITATE.PAS

```
Program parity;
const  ni='PARITY.IN';
       no='PARITY.OUT';
       nmax=5000;
type cell=record
            s,f:Longint;
            p:Byte;
          end;
var f:Text;
    aux:cell;
    a:array [1..nmax] of cell;
    n,pas,i,start,finish,l,k:Longint;
    vali:string;
```

```
procedure stop(m:longint);
begin
  Close(f);
  Assign(f,no);
  Rewrite(f);
  Writeln(f,m);
  Close(f);
  Halt
end;

procedure work;
begin
  for i:=1 to length(vali) do
    vali[i]:=upcase(vali[i]);
  l:=l+1;
  a[l].s:=start;
  a[l].f:=finish;
  if pos('EVEN',vali)>0 then
    a[l].p:=0
  else a[l].p:=1;
  k:=l-1;
  while (k>0) and (a[k].s>=a[l].s) do
    k:=k-1;
  k:=k+1;
  while (k<l) and (a[k].s=a[l].s) do
  begin
    if a[k].f>a[l].f then
    begin
      aux:=a[k];
      a[k]:=a[l];
      a[l]:=aux
    end;
    if a[k].f=a[l].f then
    begin
      if a[k].p<>a[l].p then
        stop(pas-1);
      l:=l-1;
      k:=l
    end;
    if a[k].f<a[l].f then
    begin
      a[l].s:=a[k].f+1;
      a[l].p:=abs(a[l].p-a[k].p)
    end;
    while (k<l) and (a[k].s<a[l].s) do
      k:=k+1
  end;
  while k<l do
  begin
    aux:=a[k];
    a[k]:=a[l];
    a[l]:=aux;
    k:=k+1
  end
end;
```

```
procedure load;
begin
  l:=0;
  Assign(f,ni);reset(f);
  Readln(f);
  Readln(f,n);
  for pas:=1 to n do
  begin
    Read(f,start,finish);
    Readln(f,vali);
    work
  end;
  Close(f);
  stop(n)
end;

Begin
  load
End.
```

P010004: Vizita orașului

Aceasta este o problemă clasică de grafuri: determinarea ciclului de lungime minimă într-un graf neorientat.

Observăm mai întâi că, dacă între două noduri sunt mai multe muchii, nu vom folosi niciodată două dintre ele în alcătuirea ciclului cerut (pentru că dacă le-am folosi, ar însemna ca ciclul să fie alcătuit numai din acele două noduri deoarece trebuie să fie elementar; dar acest lucru contravine condiției din enunț `k>2` - ciclul trebuie să fie format din cel puțin trei noduri). Deducem deci că, dacă între două noduri există mai multe muchii o reținem doar pe cea de cost minim.

Vom determina întâi ciclul de lungime minimă care conține neapărat nodul `1`, apoi ciclul de lungime minimă care conține neapărat nodul `2` și vom continua până la nodul `N`. Minimul acestor cicluri va constitui, evident, soluția problemei.

Cum determinăm ciclul de lungime minimă care trece prin nodul `I`? Considerând un nod arbitrar `J` de pe acest ciclu, vom putea scrie ciclul ca reuniune a două drumuri de la `I` la `J`, drumuri ce nu au în comun decât nodurile `I` și `J`. Evident, unul dintre aceste drumuri are costul egal cu cel al drumului minim de la `I` la `J` (altfel, ar putea fi substituit cu acesta, costul ciclului minim devenind mai mic). Celălalt drum este *al doilea drum minim* care are costul cel puțin egal cu al primului. Considerând acest al doilea drum ca o reuniune de drumuri (fie numărul acestora `K`), el va fi format din `K-1` drumuri optime (minime) și un drum de cost cel puțin egal cu al drumului optim corespunzător. Ideea ce se conturează de aici este următoarea: aplicăm algoritmul *Dijkstra*, având ca nod sursă nodul `I`. Acest algoritm determină drumurile minime de la nodul `I` la celelalte noduri, drumuri ale căror reuniune formează un arbore cu rădăcina `I` (acest lucru este evident - legăturile de tip "tată" care sunt construite de acest algoritm generează, la reconstituire, un arbore). În afara arborelui

creat ramân, evident, M-(N-1)=M-N+1 muchii (unde M era numărul inițial de muchii). Se consideră oricare asemenea muchie și ciclul format de ea împreună cu drumurile minime de la capetele ei până la rădăcina arborelui. În cazul în care costul unui asemenea ciclu este mai mic decât costul ciclului optim, atunci acesta este actualizat.

Procedura Load citește fișierul de intrare și reține, în matricea A, pentru fiecare pereche de noduri muchia de cost minim dintre ele.

Procedura Dijkstra implementează algoritmul *Dijkstra* având un nod sursă.

Procedura Actua calculează costul fiecărui ciclu format dintr-o muchie neinclusă în arborele rezultat și drumurile optime de la capetele ei la nodul sursă.

Procedura principală este Work, care apelează procedura Dijkstra considerând fiecare nod pe rând ca fiind nod sursă.

În cazul în care graful format doar din muchiile de cost minim este un arbore, problema nu admite soluție, caz detectat de procedura Imposibil. În orice alt caz, se apelează procedura Reconst, care reconstituie ciclul de lungime minimă, afișat apoi de procedura Scrie.

Din fiecare nod, este aplicat algoritmul *Dijkstra*, care are complexitate pătratică. De asemenea, numărul muchiilor rămase este cel mult proporțional cu N. Așadar complexitatea algoritmului este $O(N^3)$. Cum limita lui N este 100, programul rulează în timp pentru orice test.

Listing VIZITA.PAS

```
program trip;

const  ni='TRIP.IN';
       no='TRIP.OUT';
       nmax=100;
       huge=2000000000;

var f:text;
    a:array [1..nmax,1..nmax] of
                                Longint;
    d,s,t,d1,d2:array [1..nmax] of
                                Longint;
    n,m,i,j,k,h,minim,min,sursa,ns,n1,
                    n2,l1,l2:longint;

procedure load;
begin
  Assign(f,ni);
  Reset(f);
  Readln(f,n,m);
  for i:=1 to n do
    for j:=1 to n do
      a[i,j]:=huge;
  for h:=1 to m do
  begin
    Readln(f,i,j,k);
    if k<a[i,j] then
    begin
      a[i,j]:=k;
      a[j,i]:=k
    end
  end;
  Close(f)
end;

procedure init;
begin
  minim:=huge;
end;

procedure dijkstra;
begin
  for i:=1 to n do
  begin
    d[i]:=a[sursa,i];
    if d[i]<>huge then t[i]:=sursa
    else t[i]:=0;
    s[i]:=0
  end;
  s[sursa]:=1;
  for i:=1 to n-1 do
  begin
    min:=huge;
    for j:=1 to n do
      if s[j]=0 then
        if d[j]<min then
        begin
          min:=d[j];
          k:=j
        end;
    if min<>huge then
    begin
      s[k]:=1;
      for j:=1 to n do
        if s[j]=0 then
          if a[k,j]<>huge then
            if min+a[k,j]<d[j] then
            begin
              d[j]:=min+a[k,j];
              t[j]:=k
            end
    end
  end;
  t[sursa]:=0
end;

procedure actua;
begin
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if a[i,j]<>huge then
```

```
          if (t[i]<>j)and(t[j]<>i) then
            if d[i]<>huge then
              if d[j]<>huge then
                if a[i,j]+d[i]+d[j]
                                <minim then
                begin
                  minim:=a[i,j]+
                                d[i]+d[j];
                  ns:=sursa;
                  n1:=i;
                  n2:=j
                end
end;

procedure work;
begin
  for sursa:=1 to n do
  begin
    dijkstra;
    actua
  end
end;

procedure imposibil;
begin
  if minim=huge then
  begin
    Assign(f,no);rewrite(f);
    Writeln(f,'No solution.');
    Close(f);
    Halt
  end
end;

procedure reconst;
begin
  sursa:=ns;
  dijkstra;
  l1:=0;
  while n1<>sursa do
  begin
    l1:=l1+1;
    d1[l1]:=n1;
    n1:=t[n1]
  end;
  l1:=l1+1;
  d1[l1]:=sursa;
  l2:=0;
  while n2<>sursa do
  begin
    l2:=l2+1;
    d2[l2]:=n2;
    n2:=t[n2]
  end
end;
```

```
procedure scrie;
begin
  Assign(f,no);rewrite(f);
  for i:=1 to l1 do
    Write(f,d1[i],' ');
  for i:=l2 downto 1 do
    Write(f,d2[i],' ');
  Writeln(f);
  Close(f)
end;

Begin
  load;
  init;
  work;
  imposibil;
  reconst;
  scrie
End.
```

P010005: Pe tabla de șah

La prima citire a textului, metoda de rezolvare cea mai indicată pare a fi *Branch & Bound*. Însă această metodă are deficiențele ei: este destul de greu implementabilă, iar programul rezultat consuma multă memorie (nemaivorbind de faptul că este posibil să nu se încadreze în timpul de execuție).

Dacă însă estimăm numărul total de configurații posibile, remarcăm că acesta este egal cu numărul de submulțimi de 8 elemente ale unei mulțimi de 16 elemente, altfel spus: combinări de 16 luate câte 8, adică 12.870.

Mai remarcăm, de asemenea, că fiecare configurație poate fi codificată printr-un număr, reprezentând echivalentul zecimal al numărului binar obținut prin liniarizarea ei. Algoritmul de rezolvare începe să se contureze: va fi o "expandare", bazată pe ideea din algoritmul lui Lee. Vom folosi o structură de date de tip coadă, inițializată cu starea inițială. La fiecare pas, vom adăuga toate configurațiile care se pot obține din configurația curentă și care nu au fost obținute încă. Astfel, fiecare configurație este obținută într-un număr minim de pași. Procesul se termină la obținerea configurației finale (evident, întotdeauna există soluție - mai mult, se poate demonstra ușor că ea se obține în cel mult 24 de pași).

Pentru o mai bună înțelegere a rezolvării, se pot considera cele 12.870 de configurații posibile ca fiind nodurile unui graf. Între două asemenea noduri există muchie de cost unitar dacă ele reprezintă configurații ce se pot obține una din alta printr-o singură mutare. Algoritmul implementeaza astfel o parcurgere BF (în lățime) a grafului, pornind din nodul ce reprezintă starea inițială și oprindu-se pe nivelul ce include nodul ce reprezintă starea finală.

Procedura Load citește stările inițială și finală.

Procedura Egale tratează cazul banal, în care cele două stări sunt identice, caz în care scrie în fișierul de ieșire 0 (numărul de mutări) și suspendă execuția programului.

Explicația procedurilor Board_To_Number și Number_To_Board este puțin mai delicată. Am considerat toate cele 12.870 configurații posibile, sortate descrescător după echivalentul zecimal al numărului binar obținut prin liniarizarea unei asemenea configurații și am asociat fiecărei configurații indicele corespunzător ei în această sortare. Astfel, procedura Board_To_Number primește o configurație și returnează numărul ei de ordine în șirul configurațiilor sortate; procedura Number_To_Board primește un număr (cuprins între 1 și 12.870) și returnează configurația aflată la indicele respectiv. Ambele proceduri folosesc matricea C, determinată de procedura Calc_Comb; C[i,j] are valoarea combinari de i luate câte j; metoda de determinare a matricei C este triunghiul lui *Pascal*.

Procedura Init inițializează coada V cu configurația de start. Păstrăm de asemenea un vector caracteristic S (care indică dacă o configurație a fost obținută sau nu), precum și un vector T, care arată pentru fiecare configurație anterioară ei (configurația din care a fost obținută).

Procedura Work completează coada până la găsirea soluției. Ea folosește procedura Move care, primind o configurație, generează toate configurațiile care se pot obține din ea printr-o mutare.

După găsirea soluției se apelează procedura Reconst care, utilizând vectorul T, reconstituie drumul de ajungere din configurația inițială în cea finală. Soluția este scrisă în fișierul de ieșire de procedura Scrie.

Fiecare configurație poate fi obținută din cel mult alte 24 de configurații (numărul de interschimbări de celule vecine într-o matrice de dimensiuni 4×4). Se poate deci considera complexitatea de ordinul *O(Numărul total de configurații)*.Acest număr este o constantă (12.870), deci complexitatea este *O(1)*. Sursa rulează aproape instantaneu pe orice test.

Listing SAH.PAS

```
Program game;
const ni='GAME.IN';
      no='GAME.OUT';
type  board=array [1..4,1..4] of Word;
      vector=array [1..12870] of Word;
      pvector=^vector;
var f:Text;
    c:array [1..16,0..16] of Word;
    v,t:vector; s:pvector;
    drum:array [1..48] of Word;
    n,i,j,k,codini,codfin,p,u,l,aux:
                                  Word;
    a,ini,fin,now,next:board;
    ch:Char; eg,found:boolean;

procedure load;
begin
  Assign(f,ni);
  Reset(f);
  for i:=1 to 4 do
  begin
    for j:=1 to 4 do
    begin
      Read(f,ch);
      if ch='1' then ini[i,j]:=1
      else ini[i,j]:=0
    end;
    Readln(f)
  end;
  Readln(f);
  for i:=1 to 4 do
  begin
    for j:=1 to 4 do
    begin
      Read(f,ch);
      if ch='1' then fin[i,j]:=1
      else fin[i,j]:=0
    end;
    Readln(f)
  end;
  Close(f)
end;

procedure egale;
begin
  eg:=true;
  for i:=1 to 4 do
    for j:=1 to 4 do
      if ini[i,j]<>fin[i,j] then
        eg:=false;
  if eg then
  begin
    Assign(f,no); Rewrite(f);
    Writeln(f,0);
    Close(f); Halt
  end
end;

procedure calc_comb;
begin
  c[1,0]:=1; c[1,1]:=1;
  for i:=2 to 16 do
  begin
    c[i,0]:=1; c[i,i]:=1;
    for j:=1 to i-1 do
      c[i,j]:=c[i-1,j-1]+c[i-1,j]
  end
end;

procedure board_to_number(a:board;
                          var n:Word);
var i,j,s,u:Word;
begin
  s:=16;
```

```
  u:=8;
  n:=0;
  for i:=1 to 4 do
    for j:=1 to 4 do
    begin
      if a[i,j]=0 then
        n:=n+c[s-1,u-1]
      else u:=u-1;
      s:=s-1;
      if u=0 then
      begin
        j:=4;
        i:=4
      end
    end;
  n:=n+1
end;

procedure number_to_board(n:Word;
                          var a:board);
var i,j,s,u:Word;
begin
  s:=16;
  u:=8;
  for i:=1 to 4 do
    for j:=1 to 4 do
      a[i,j]:=0;
  for i:=1 to 4 do
    for j:=1 to 4 do
      if not ((i=4) and (j=4)) then
      begin
        if n>c[s-1,u-1] then
          n:=n-c[s-1,u-1]
        else
        begin
          a[i,j]:=1;
          u:=u-1
        end;
        s:=s-1;
        if u=0 then
        begin
          j:=4;
          i:=4;
        end
      end
      else
        if n>0 then a[i,j]:=1
        else a[i,j]:=0
end;

procedure init;
begin
  New(s);
  for i:=1 to c[16,8] do
    s^[i]:=0;
  board_to_number(ini,n);
  v[1]:=n;
  t[1]:=0;
  s^[n]:=1;
  p:=1;
  u:=1;
  board_to_number(ini,n);
  codini:=n;
  board_to_number(fin,n);
  codfin:=n
end;

procedure move;
begin
  for i:=1 to 4 do
    for j:=1 to 3 do
      if a[i,j]<>a[i,j+1] then
      begin
        a[i,j]:=1-a[i,j];
        a[i,j+1]:=1-a[i,j+1];
        board_to_number(a,n);
        a[i,j]:=1-a[i,j];
        a[i,j+1]:=1-a[i,j+1];
        if s^[n]=0 then
        begin
          s^[n]:=1; u:=u+1;
          v[u]:=n; t[u]:=p;
          if n=codfin then
          begin
            found:=true;
            j:=3; i:=4
          end;
        end;
      end;
  if not found then
    for i:=1 to 3 do
      for j:=1 to 4 do
        if a[i,j]<>a[i+1,j] then
        begin
          a[i,j]:=1-a[i,j];
          a[i+1,j]:=1-a[i+1,j];
          board_to_number(a,n);
          a[i,j]:=1-a[i,j];
          a[i+1,j]:=1-a[i+1,j];
          if s^[n]=0 then
          begin
            s^[n]:=1; u:=u+1;
            v[u]:=n; t[u]:=p;
            if n=codfin then
            begin
              found:=true; j:=4; i:=3
            end
          end
        end
end;
```

```
procedure work;
begin
  found:=false;
  while (p<=u) and (not found) do
  begin
    n:=v[p];
    number_to_board(n,a);
    move;
    if not found then p:=p+1
    else write
  end
end;

procedure reconst;
begin
  l:=0;
  while u>0 do
  begin
    l:=l+1; drum[l]:=v[u];
    u:=t[u]
  end;
  p:=1; u:=l;
  while p<u do
  begin
    aux:=drum[p];
    drum[p]:=drum[u];
    drum[u]:=aux;
    p:=p+1; u:=u-1
  end
end;

procedure scrie;
begin
  Assign(f,no); Rewrite(f);
  Writeln(f,l-1);
  for k:=1 to l-1 do
  begin
    n:=drum[k];
    number_to_board(n,a);
    now:=a; n:=drum[k+1];
    number_to_board(n,a);
    next:=a;
    for i:=1 to 4 do
      for j:=1 to 4 do
        if now[i,j]<>next[i,j] then
          write(f,i,' ',j,' ');
    Writeln(f)
  end;
  Close(f)
end;

Begin
  Load;
  egale;
  calc_comb;
  init;
  work;
  reconst;
  scrie
End.
```

P010006: Jocul de-a constructorul

Aceasta a fost una dintre cele mai frumoase probleme din concurs, problema aparținând unui gen cunoscut sub numele de "problemă de idee", necesitând multă imaginație, vedere în spațiu și, nu în ultimul rând, cunoștințe avansate de prelucrarea structurilor mari de date. Programul rezultat este foarte scurt, însă raționamentele care stau în spatele lui sunt migăloase și, sperăm, interesante de urmărit.

Mai întâi, trebuie stabilit când problema admite soluție și când nu. Pentru aceasta, să considerăm un plan arbitrar de oraș; evident, el va avea cel puțin un turn de înălțime maximă. Atunci acel turn va fi, în mod clar, atât maximul turnurilor privind din față, cât și din lateral dreapta. În consecință, o condiție necesară pentru ca planul să fie realizabil este că maximele celor două șiruri să fie egale. În continuare, se va demonstra în mod constructiv că această condiție este și suficientă.

Considerând îndeplinită această condiție, să construim orașul cu un număr minim de blocuri. Notăm cu `L` linia pe care, la vederea din dreapta, se atinge maximul înălțimilor (dacă sunt mai multe, una dintre ele); analog, notăm cu `C` coloana pe care, la vederea din față, se atinge maximul înălțimilor (dacă sunt mai multe, una dintre ele). Pentru a satisface cerințele vederii din față, așezăm pe linia `L`, corespunzător fiecărei coloane, câte un turn având înălțimea citită. Astfel condițiile vederii din față sunt îndeplinite folosind, evident, un număr minim de blocuri (suma elementelor ce alcătuiesc primul șir citit). Trecem acum la cerințele vederii din lateral dreapta. Condiția pe linia `L` este satisfăcută (se vede turnul de înălțime maximă). Parcurgem acum restul liniilor. Să presupunem că pe o linie ni se cere un turn de înălțime `H`, căutăm înălțimea `H` printre cerințele vederii din față. Dacă o găsim, notăm cu `CL` coloana pe care se află și "translatăm" turnul respectiv pe coloana `CL` până în dreptul liniei cerute. Astfel, satisfacem cerința vederii din lateral, iar privind din față, nu s-a schimbat nimic (turnul a "culisat" în lungul unei coloane, deci nu se vede nici o schimbare). Dacă, însă, nu găsim înălțimea `H` printre cerințele vederii din față, suntem nevoiți să construim un turn de înălțime `H`, pe care îl plasăm pe linia a cărei condiție o îndeplinim în momentul respectiv și pe coloana `C`, fiind astfel "mascat" la privirea din față de turnul de înălțime maximă. În acest al doilea caz, numărul total de blocuri necesare se incrementează cu `H`.

Să construim acum orașul cu număr maxim de blocuri. Din nou, vom satisface întâi cerințele vederii din față. De data aceasta însă, vom parcurge coloanele și vom pune pe fiecare linie câte un turn de înălțimea cerută. Calculăm numărul de blocuri folosite ca fiind suma elementelor primu-

lui șir citit înmulțită cu numărul de linii al matricei (fiecare turn este pus pe fiecare componentă a coloanei pe care stă el). Însă dacă privim acum din lateral, vedem pe fiecare linie câte un turn de înălțime maximă. Parcurgem deci liniile și, pe fiecare linie, "tăiem" tot ce depășește înălțimea cerută respectivei linii. Notând cu `HMAX` maximul înălțimilor și `H` - înălțimea cerută liniei curente, vom "tăia" `(HMAX - H)*` *numărul de coloane al matricei blocuri*, număr ce va trebui scăzut din numărul inițial de blocuri folosite.

În program, față de explicațiile prezentate aici, apare o mică modificare și anume vectorii înălțimilor nu sunt reținuți așa cum sunt citiți ci sunt reținuți doi vectori `A` și `B`, în care `A[i]`, respectiv `B[i]`, indică de câte ori este cerută înălțimea `i` la vederea din față, respectiv de câte ori este cerută înălțimea `i` la vederea din dreapta. Acest lucru este posibil, pentru că, evident, ordinea liniilor și coloanelor nu contează, și este necesar, pentru că matricea are dimensiunile maxime `100.000×100.000` (ceea ce este evident prea mult) dar maximul înălțimilor este doar `5.000`.

Procedura `Load` citește vectorii înălțimilor, calculează maximele lor și contruiește vectorii `A` și `B`.

Procedura `Imposibil` detectează cazul fără soluție, ieșind forțat după scrierea unui mesaj corespunzător.

Procedurile `Minim` și `Maxim` calculează valorile cerute, notate cu `MIN` și `MAX`, după algoritmii descriși mai sus.

Rezultatele sunt afișate de procedura `Scrie`.

Cea mai mare parte a timpului de rulare este consumată de citire; prelucrările ulterioare sunt proporționale cu `HMAX`, care este cel mult `5.000`. Așadar, ordinul de complexitate este *O (Input + HMAX)*, adică minimul posibil. Programul trece orice test, încadrându-se fără probleme în timpul de execuție admis.

Listing ORAS.PAS

```
Program town;
const  ni='TOWN.IN';
       no='TOWN.OUT';
       hmax=5000;
var f:Text;
    a,b,c:array [0..hmax] of Longint;
    n,m,i,j,k,maxa,maxb,min,max:Longint;

procedure load;
begin
  Assign(f,ni); Reset(f);
  Readln(f,n,m);
  for i:=0 to hmax do
  begin a[i]:=0; b[i]:=0 end;
  maxa:=0;
  for i:=1 to n do
  begin
    Readln(f,k);
    a[k]:=a[k]+1;
    if k>maxa then maxa:=k
  end;
  maxb:=0;
  for i:=1 to m do
  begin
    Readln(f,k); b[k]:=b[k]+1;
    if k>maxb then maxb:=k
  end;
  Close(f)
end;

procedure imposibil;
begin
  if maxa<>maxb then
  begin
    Assign(f,no); Rewrite(f);
    Writeln(f,'No solution.');
    Close(f); Halt
  end
end;

procedure minim;
begin
  min:=0;
  for i:=0 to hmax do min:=min+i*a[i];
  for i:=0 to hmax do
    if b[i]>a[i]
     then min:=min+i*(b[i]-a[i])
end;

procedure maxim;
begin
  max:=0;
  for i:=0 to hmax do
    max:=max+i*a[i]*m;
  c[hmax]:=0; j:=0;
  for i:=hmax-1 downto 0 do
  begin
    j:=j+a[i+1]; c[i]:=c[i+1]+j
  end;
  for i:=0 to hmax do
    max:=max-c[i]*b[i]
end;

procedure scrie;
begin
  Assign(f,no);
  Rewrite(f);
  Writeln(f,min,' ',max);
  Close(f)
end;

Begin
  load; imposibil;
  minim; maxim;
  scrie
End.
```

CONCURS

Se pare că problemele de la a cincea rundă a concursului nostru au fost foarte ușoare, majoritatea concurenților obținând peste 200 de puncte. Vă prezentăm în continuare soluțiile acestor probleme, precum și clasamentul după cinci etape.

Ginfo Top 30

1.	Adrian Cârcu, Bistrița	1328
2.	Mihai Pătraşcu, Craiova	1306
3.	Cristian Cadar, Bucureşti	1230
4.	Ioana Ileană, Alba Iulia	1158
5.	Daniel Dumitran, Bucureşti	1113
6.	Mihai Stroe, Bucureşti	1081
7.	Ciprian Baicu, Dr. Tn. Severin	1076
8.	Vlad Vâlceanu, Arad	1051
9.	Csaba Patcas, Oradea	1003
10.	David Andrei, Râmnicu Vâlcea	983
11.	Gabriel Erzse, Oradea	978
12.	Gabriel Pârvan, Râmnicu Sărat	912
13.	Marian Anghel, Buftea	866
14.	Cristian Alexandrescu, Botoşani	844
15.	Virgil Şerbănuță, Bucureşti	834
16.	VictorAsavei Dr. Tn. Severin	820
17.	Silvestru Negruseri, Bistrița	807
18.	Sorin Otescu, Braşov	735
18.	Liviu Păunescu, Constanța	735
20.	Ileana Constantinescu, Sibiu	734
21.	Mihai Dondera, Bucureşti	716
22.	Horea Haitonic, Năsăud	696
23.	Mihai Maerean, Slobozia	668
24.	Ciprian Cană, Vaslui	666
25.	Lucian Burja, Blaj	647
26.	Lucian Lazăr, Câmpeni	637
27.	Marius Muja, Orăştie	618
28.	Adrian Bălan, Târgu Jiu	583
29.	Daniel Bolohan,Iaşi	566
30.	Daniel Popa, Orăştie	533

Triunghi

Mulți concurenți au considerat această problemă ca fiind cea mai dificilă de la runda a cincea. Totuși majoritatea au reușit să o rezolve corect, punctajul mediu obținut fiind de 67.02 puncte.

Problema ar fi trebuit să fie cunoscută, ea fiind spre rezolvare și în numărul 1/1998 al Gazetei de Informatică (**P019803**).

Datorită faptului că rezolvarea este foarte simplă vom trece direct la prezentarea soluției. Implementarea îi aparține lui *Vlad Lazarciuc* din *Baia Mare* (cea mai scurtă sursă corectă).

Listing TRIUNGHI.PAS

```
{$N+}
var f:Text;
    x,l,j:Extended;
    i,t:Longint;
    timer:Longint absolute $0000:$046c;

procedure nosol;
begin
  Writeln(f,'0 0 0');
  Close(f);
  Halt
end;

Begin
  t:=timer;
  Assign(f,'TRIUNGHI.IN');
  Reset(f);
  Read(f,x);
  Close(f);
  Assign(f,'TRIUNGHI.OUT');
  Rewrite(f);
  for i:=1 to Round(Sqrt(x)) do
  begin
    if timer-t>17
      then nosol;
    l:=-2*i*i+6*i+4*x-12;
    if l<0
      then continue;
    l:=sqrt(l);
    l:=int((l-2*i)/2);
    j:=(x-(l*l+2*i*l+3*i*
                    (i-1)/2+3))/3+1;
    if (Int(j)=j)and(j<=i)and(j>0) then
    begin
      Write(f,i*(i-1)/2+j:0:0,' ',
            (i+1)*(i+l-1)/2+j:0:0,' ',
             (i+1)*(i+l-1)/2+j+1:0:0);
      Close(f);
      Halt
    end
  end;
  nosol
End.
```

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
David Andrei, Râmnicu Vâlcea
Ciprian Baicu, Dr. Tn. Severin
Adrian Bălan, Târgu Jiu
Lucian Burja, Blaj
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Mihai Dondera, București
Daniel Dumitran, București
Ioana Ileană, Alba Iulia
Liviu Lalescu, Craiova
Vlad Lazarciuc, Baia Mare
Valentin Muste, Baia Mare
Silvestru Negruseri, Bistrița
Gabriel Pârvan, Râmnicu Sărat
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Dan Popovici, Craiova
Mihai Stroe, București
Virgil Șerbănuță, București

Beculețe
(Arbore parțial minim)

Aceasta a fost problema cu punctajul mediu cel mai redus, "numai" 61.82 puncte. Rezolvarea este foarte simplă, problema reducându-se la determinarea unui arbore parțial minim într-un graf complet ale cărui noduri sunt beculețele, iar costurile muchiilor sunt date de numărul pătrățelelor prin care ar trece un fir care face legătura între perechile de beculețe. Dacă două beculețe au coordonatele (x_1, y_1) și (x_2, y_2), atunci numărul de pătrățele traversate de fir este dat de formula $|x_2-x_1|+|y_2-y_1|-cmmdc(|x_2-x_1|, |y_2-y_1|)$. Vă prezentăm în continuare implementarea propusă de *Ioana Ileană* din *Alba Iulia*.

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
Marian Anghel, Buftea
Ciprian Baicu, Dr. Tn. Severin
Daniel Bolohan, Iași
Cristian Cadar, București
Adrian Cârcu, Bistrița
Arcadie Crăcan, Bălți
Mihai Dondera, București
Gabriel Erzse, Oradea
Ioana Ileană, Alba Iulia
Liviu Lalescu, Craiova
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Raluca Sauciuc, București
Mihai Stroe, București
Virgil, Șerbănuță, București

Listing BECULETE.PAS

```
Program apm;
type vect=array[1..255] of Longint;
     pvect=^vect;
     bec=record
           x,y:Longint;
         end;
     muchie=record
              b1,b2:bec;
            end;
var a:array[1..255] of pvect;
    b:array[1..255] of bec;
    dmin:array[1..255] of Longint;
    seen:array[1..255] of Boolean;
    last:array[1..255] of Longint;
    m,n,i,j,k,p,q,min,pmin,arb:Longint;
    f:Text;

function cmmdc(a,b:longint):Longint;
begin
  if b=0 then cmmdc:=a
  else cmmdc:=cmmdc(b,a mod b)
end;

procedure citire;
begin
  Assign(f,'BECULETE.IN');
  Reset(f);
  Readln(f,n);
  for i:=1 to n do
    Readln(f,b[i].x,b[i].y);
  Close(f)
end;

procedure prelucrare;
begin
  for i:=1 to n do New(a[i]);
```

```
  for i:=1 to n-1 do
    for j:=i+1 to n do
    begin
      p:=Abs(b[i].x-b[j].x);
      q:=Abs(b[i].y-b[j].y);
      a[i]^[j]:=p+q;
      if p>q then
        Dec(a[i]^[j],cmmdc(p,q))
      else Dec(a[i]^[j],cmmdc(q,p));
      a[j]^[i]:=a[i]^[j]
    end;
  for j:=1 to n do
    dmin[j]:=Maxlongint;
  k:=1;
  for i:=2 to n do
  begin
    seen[k]:=true;
    min:=Maxlongint;
    for j:=1 to n do
      if not seen[j] then
      begin
        if a[k]^[j]<dmin[j] then
        begin
          last[j]:=k;
          dmin[j]:=a[k]^[j]
        end;
        if dmin[j]<min then
        begin
          min:=dmin[j];
          pmin:=j
        end
      end;
    k:=pmin;
    arb:=arb+a[last[k]]^[k]
  end
end;

procedure afisare;
begin
  Assign(f,'beculete.out');
  Rewrite(f);
  Writeln(f,arb);
  Writeln(f,n-1);
  for i:=2 to n do
    Writeln(f,b[last[i]].x,' ',
            b[last[i]].y,' ',b[i].x,' ',
                                b[i].y);
  Close(f)
end;

Begin
  citire;
  prelucrare;
  afisare
End.
```

Punctaj maxim...

În sfârşit avem din nou concurenţi care au obţinut punctaj maxim (300 puncte) la o rundă a concursului Bursele Agora.

Chiar dacă problemele au fost foarte simple, performanţa lor este demnă de remarcat.

Cei zece concurenţi care au reuşit să rezolve corect toate cele trei probleme ale rundei a cincea sunt:

Cristian Alexandrescu, Botoşani

Ciprian Baicu, Dr. Tn. Severin

Adrian Cârcu, Bistriţa

Mihai Dondera, Bucureşti

Ioana Ileană, Alba Iulia

Liviu Lalescu, Craiova

Csaba Patcas, Oradea

Mihai Pătraşcu, Craiova

Mihai Stroe, Bucureşti

Virgil Şerbănuţă, Bucureşti

Felicitări!!!

Descalificări...

Din nefericire pentru unii redacţia GInfo a descoperit din nou surse aproape identice trimise de mai mulţi concurenţi. De această dată unul dintre "vinovaţi" se afla chiar printre primii clasaţi cu şanse foarte mari de a participa la faza finală.

Comitetul de organizare al concursului Bursele Agora a decis descalificarea următorilor concurenţi:

Silviu Bota, Baia Mare

Robert Baban, Baia Mare

Runda 5, Problema 3

Multiplu

Aceasta a fost cea mai simplă problemă propusă în cadrul concursului Bursele Agora, dovadă și punctajul mediu obținut de concurenți: 85,34 puncte.

Dacă *n* este numărul al cărui multiplu trebuie determinat și *b* cifra care poate apărea în acest multiplu (alături de 0), determinăm resturile împărțirilor la *n* a numerelor formate numai din cifra *b*: *b*, *bb*, *bbb* etc. În momentul în care obținem două resturi egale pentru două numere de această formă, scădem cele două numere și obținem un număr de forma *b...b0...0*, care conține numai cifre 0 și *b* și al cărui rest la împărțirea cu *n* este 0 (deci acest număr este multiplu al lui *n*). Vă prezentăm în continuare implementarea propusă de *Virgil Șerbănuță* din *București*.

Listing MULTIPLU.PAS

```
var resturi:array[0..15000] of Integer;
    n,b,x,y:Integer; f:Text;

procedure rezolva;
var i,k,l:Longint;
begin
  for i:=1 to 15000 do resturi[i]:=-1;
  resturi[0]:=0; k:=0; l:=1;
  for i:=1 to 15000 do
  begin
    k:=(k+l*b) mod n; l:=l*10 mod n;
    if resturi[k]>=0 then
    begin
      x:=resturi[k]; y:=i; Break
    end;
    resturi[k]:=i
  end;
end;

procedure afisare;
var i:Integer;
begin
  for i:=y downto 1 do
    if i>x then Write(f,b)
    else Write(f,0)
end;

Begin
  Assign(Input,'MULTIPLU.IN');
  Reset(Input);
  Assign(f,'MULTIPLU.OUT'); Rewrite(f);
  Readln(n,b);
  rezolva;
  afisare;
  Close(f)
End.
```

Au rezolvat corect:

Cristian Alexandrescu, Botoşani
David Andrei, Râmnicu Vâlcea
Marius George Andrei, Bucureşti
Marian Anghel Buftea
Victor Asavei, Dr. Tn. Severin
Ciprian Baicu, Dr. Tn. Severin
Robert Bădiță, Craiova
Adrian Bălan, Târgu Jiu
Daniel Bolohan, Iaşi
Lucian Burja, Blaj
Cristian Cadar, Bucureşti
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Arcadie Crăcan, Bălți
Mihai Dondera, Bucureşti
Daniel Dumitran, Bucureşti
Gabriel Erzse, Oradea
Horea-Adrian Haitonic, Năsăud
Ioana Ileană, Alba Iulia
Liviu Lalescu, Craiova
Codruț-Lucian Lazăr, Câmpeni
Mihai Maerean, Slobozia
Silviu Maniu, ***
Marius Muja, Orăştie
Sorin Otescu, Braşov
Gabriel Pârvan, Râmnicu Sărat
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Liviu Păunescu, Constanța
Daniel Popa, Orăştie
Raluca Sauciuc, Buxureşti
Mihai Stroe, Bucureşti
Virgil Șerbănuță, Bucureşti

BACALAUREAT 2000

Prezentarea subiectelor de Bacalaureat propuse în cartea Bac Info' 2000 continuă cu varianta nr. 19.

Subiectul 1

1. Se consideră funcția `caut` care ar trebui să returneze `true` dacă numărul `nr` se găsește în șirul `x`. În variabila `k` se va păstra poziția pe care s-a găsit numărul căutat sau -1 dacă numărul nu se găsește în șir. Găsiți greșeala și scrieți funcția corectă:

Varianta Pascal:

```
var x:array[1..10] of Real;
function caut(nr:Real; var k:Byte):Boolean;
begin
  caut:=false;
  k:=-1;
  for i:=1 to 10 do
  begin
    caut:=x[i]=nr;
    k:=i
  end
end;
```

Varianta C:

```
float x[10];
int caut(float nr,int& k){
  int c=0;
  k=-1;
  for (int i=0;i<10;i++)
    c=(x[i]==nr),k=i;
  return c;
}
```

Soluție

Greșeala constă în faptul că datorită structurii repetitive **for**, în funcție, practic, se suprascrie valoarea identificatorului `caut` și a variabilei `k`. Rezultatul va fi corect doar dacă numărul căutat este chiar al `n`-lea, adică ultimul.

O posibilă variantă corectă trebuie să asigure oprirea căutării dacă numărul căutat este găsit și tratarea separată a cazului căutării fără succes. Deoarece în funcție nu putem testa valoarea lui `caut` (ar însemna un autoapel), avem nevoie de o variabilă auxiliară (`gasit`).

În cazul variantei *Pascal* mai apare o greșeală "minoră", și anume: `k` este declarat de tip `Byte`, dar printre valorile posibile pe care această variabilă le poate lua este și -1. Rezultă că trebuie să fie declarat de tipul `Shortint` sau `Integer`, tipuri care admit și valori negative. O funcție corectă este prezentată în continuare.

Varianta Pascal:

```
var x:array[1..10] of Real;
function caut(nr:Real; var k:Shortint):
                                    Boolean;
var i:Byte;
    gasit:Boolean;
begin
  gasit:=false; k:=-1; i:=1;
  while (i<=n) and not gasit do
    if nr=x[i]
    then begin gasit:=true; k:=i end
    else i:=i+1;
 caut:=gasit
end;
```

Varianta C:

```
float x[10];
int caut(float nr,int& k){
  int c=0,gasit=0;
  k=-1;
  while (i<n && !gasit)
    if (nr==x[i])
      gasit=1,k=i;
    else i++;
  return gasit;
}
```

2. Scrieți un subprogram care calculează și afișează media aritmetică a elementelor pozitive, diferite de 0 de pe diagonala principală, respectiv diagonala secundară a unui tablou bidimensional pătratic de numere întregi.

Soluție

Elementele pe diagonala principală au indicele de linie egal cu indicele de coloană, iar cele de pe diagonala secundară

sunt de forma i,n-i+1 (în cazul în care numerotarea indicilor începe de la 1 - limbajul *Pascal*), respectiv i,n-i-1 (în cazul în care numerotarea indicilor începe de la 0 - limbajul C). De fiecare dată, când un element se adaugă sumei s, trebuie mărit contorul k, deoarece pentru a putea calcula media aritmetică trebuie să știm câte elemente au format suma.

Varianta Pascal:

```
const DimMax=100;
type tab=array[1..DimMax,1..DimMax] of
                                    Integer;
var a:tab;
    i,j,n:Byte;

procedure calcule(a:tab; n:Byte);
var i,j,k:Byte;
    s:Real;
begin
  s:=0; k:=0;
  for i:=1 to n do
  begin
    if a[i,i]>0
    then begin s:=s+a[i,i]; Inc(k) end;
    if a[i,n-i+1]>0
    then begin s:=s+a[i,n-i+1]; Inc(k) end
  end;
  s:=s/k;
  Writeln('Media aritmetica: ',s:10:2)
end;
```

Varianta C:

```
#define DimMax 100;
typedef int tab[DimMax][DimMax];
tab a;
int i,j,n;

void calcule(tab a, int n){
  int i,j,k;
  float s=k=0;
  for (i=0;i<n;i++){
    if (a[i][i]>0)
      s+=a[i][i],k++;
    if (a[i][n-i-1]>0)
      s+=a[i][n-i-1],k++;
  }
  s/=k;
  printf("Media aritmetica: %10.2f\n",s);
}
```

3. Se consideră următoarea secvență de instrucțiuni, reprezentată în pseudocod. Ce valoare se va afișa pentru m=64?

```
citește m
d ← 2
cât timp m ≠ 0 execută
  dacă rest[m/d] = 0 atunci scrie d
                     altfel d ← d + 1
  sfârșit dacă
  m ← [m/d]
sfârșit cât timp
```

Transcrieți această secvență de program într-un limbaj de programare (*Pascal* sau C).

Soluție

Secvența de algoritm afișează factorii primi ai numărului citit (m). Pentru m=64 se va afișa 2 2 2 2 2 2. Implementările *Pascal* și C ale algoritmului sunt prezentate în continuare:

Varianta Pascal:

```
Program factori_primi;
var m,d:Byte;
Begin
  Write('m=');
  Readln(m);
  d:=2;
  while m<>0 do
  begin
    if m mod d=0 then Writeln(d)
                 else Inc(d);
    m:=m div d
  end
End.
```

Varianta C:

```
#include <stdio.h>
void main(void){
  int m,d=2;
  printf("m=");
  scanf("%d",&m);
  while (m){
    if (!(m%d)) printf("%d\n",d)
    else d++;
    m/=d;
  }
}
```

Subiectul 2

1. Se numesc numere "superprime" acele numere prime ale căror prefixe sunt toate numere prime. De exemplu, numărul 239 este superprim, deoarece numerele 2, 23 și 239 sunt prime, dar numărul 241 nu este superprim, deoarece chiar dacă 241 și 2 sunt prime, numărul 24 nu este prim.

   Scrieți un program comentat, care stabilește dacă numărul n, citit de la tastatură, este superprim sau nu! Programul va trebui să afișeze și prefixele numărului dat.

Soluție

Varianta Pascal:

Listing SUPER.PAS

```
Program superprime;
var n,nn, {    nn este dublura lui n, }
          {   deoarece se va distruge }
          { pe parcursul algoritmului }
    putere:Longint; { puterile lui 10 }
          { cu care il impartim pe n }
    superprim:Boolean;    { variabila }
              { logica, purtatoare a }
              { proprietatii cautate }

function prim(n:Longint):Boolean;
 { se verifica proprietatea de "prim" }
 {         pentru n si prefixele sale }
var d,           {  divizori posibili }
    rad:Longint; { vom cauta divizori }
    { doar pana la parte intreaga din }
    {                   radical din n }
    e_prim:Boolean; { variabila loca- }
    {    la purtatoare a proprietatii }
    {    cautate (n actual e prim sau }
    {  nu), valoarea sa se va atribui }
    {       lui prim, identificatorul }
    {                         functiei }
begin
  if not Odd(n)   { daca numarul este }
    { par, atunci el este prim numai }
    {           daca are valoarea 2 }
  then e_prim:=n=2
  else
  begin
    { initializarea variabilei locale }
    e_prim:=true;
    d:=3; { primul divizor posibil al }
          {      unui numar impar <>1 }
    rad:=Trunc(Sqrt(n));
    { parte intreaga din radical de n }
    while (d<=rad) and e_prim do
    { cautam divizori pana la radical }
    { din n, sau pana cand gasim unul }
      if n mod d=0 then e_prim:=false
      else Inc(d,2) { divizorii posi- }
        { bili ai unor numere impare }
        {          sunt numere impare }
  end;
  prim:=e_prim
end;

Begin
  Write('n='); Readln(n);
  superprim:=true;
  { presupunem ca numarul e superprim }
  nn:=n;              { dublura lui n }
  putere:=10; { prima putere a lui 10 }
    { pentru a obtine ultimul prefix }
    {                    al lui n }
  while (n>0) and superprim do
  begin
          { verificam daca n actual }
          {        este prim sau nu }
    if not prim(n)
    then superprim:=false  { daca nu, }
            { n dat nu e superprim }
    else
    begin
      n:=nn div putere;
                  { urmatorul prefix }
      Writeln(n);
              { afisarea prefixului }
      putere:=putere*10
        { urmatoarea putere al lui 10 }
    end
  end;
  if superprim then
    Writeln('Numarul ',nn,
                  ' este superprim.')
  else Writeln('Numarul ',nn,
                ' nu este superprim.')
End.
```

Varianta C:

Listing SUPER.CPP

```
#include <stdio.h>
#include <math.h>

long n,nn; // nn este dublura lui n,
           // deoarece se va distruge
           // pe parcursul algoritmului
long putere; // puterile lui 10, cu
             // care impartim pe n
int superprim; // variabila logica,
               // purtatoare a pro-
               // prietatii cautate
int prim(long n){ // se verifica pro-
  // prietatea de "prim" pentru n si
  // prefixele sale
  long d=3; // divizori posibili - 3
            // este primul divizor
            // posibil al unui numar
            // impar diferit de 1
  long rad=floor(sqrt(n)); // vom cauta
       // divizori doar pana la parte
       // intreaga din radical din n
  int e_prim=1; // variabila locala,
       // purtatoare a proprietatii
       // cautate (n este prim sau nu)
```

```
  if (!(n&1)) // daca numarul este
              // par, atunci el este
              // prim numai daca are
              // valoarea 2
    e_prim=n==2;
  else{
    while (d<=rad && e_prim){
      // cautam divizori pana la
      // radical din n, sau pana cand
      // gasim unul
      if (!(n%d)) e_prim=0;
      else d+=2; // divizorii posibili
                 // ai unor numere
                 // impare sunt numere
                 // impare
    }
  }
  return e_prim;
}

void main(void){
  printf("n=");
  scanf("%d",&n);
  superprim=1; // presupunem ca
               // numarul e superprim
  nn=n; // dublura lui n
  putere=10; // prima putere a lui 10,
             // pentru a obtine ul-
             // timul prefix al lui n
  while (n>0 && superprim){
    if (!prim(n)) // verificam daca n
                  // actual e prim sau nu
      superprim=0; // daca nu, n dat
                   // nu e superprim
    else{
      n=nn/putere; // urmatorul prefix
      printf("%d\n",n); // afisarea
                        //  prefixului
      putere*=10; // urmatoarea putere
                  // a lui 10
    }
  }
  if (superprim)
    printf("Numarul %d este \
                    superprim.\n",nn);
  else printf("Numarul %d nu este \
                    superprim.\n",nn);
}
```

2. Scrieți un subprogram `cerc` care leagă ultimul element de primul element într-o listă liniară, având o structură oarecare. Subprogramul va avea ca parametri de intrare pointerii la primul, respectiv la ultimul element din listă. Scrieți un alt subprogram care, după operația cerută mai sus, va afișa conținutul listei.

Soluție

Varianta Pascal:

```
type lista=^element;
     element=record
               info:Byte;
               leg:lista
             end;
var primul,ultimul:lista;

procedure cerc(primul:lista;
                          var ultimul:lista);
begin
  ultimul^.leg:=primul
end;

procedure afiseaza(primul,ultimul:lista);
var p:lista;
begin
  p:=primul;
  while p<>ultimul do
                { avansam pana la ultimul }
  begin
    Writeln(p^.info);
    p:=p^.leg
  end;
  Writeln(ultimul^.info)
     { ultimul element se afiseaza separat }
end;
```

Varianta C:

```
struct lista{
  int info;
  lista* leg;
}primul,ultimul:lista;

void cerc(lista* primul, lista* &ultimul){
  ultimul->leg:=primul;
}

void afiseaza(lista *primul, lista *ultimul){
  lista *p=primul;
  while (p!=ultimul){
                  // avansam pana la ultimul
    printf("%d\n",p->info);
    p=p->leg;
  }
  printf("%d\n",ultimul->info);
    // ultimul element se afiseaza separat }
}
```

Subiectul 3

Se consideră `n` (`1<n<10`) operatori din mulțimea `{+,*}` și un număr natural `val` (`val>100`). Să se scrie un program comentat, care, folosind numere naturale, pune între

fiecare doi operatori (respectiv înainte și după aceștia) câte un număr, astfel încât rezultatul evaluării să fie numărul val.

Soluție

Expresia va fi împărțită în termeni (separați prin +), iar termenii vor fi împărțiți în factori (separați prin *). Vom alege valoarea primului termen ca fiind val (valoarea expresiei), valorile celorlalți termeni (dacă aceștia există) vor fi 0. Pentru ca valoarea primului termen să fie val, valoarea primului factor al acestui termen va fi val, iar valorile celorlalți factori (dacă aceștia există) vor fi 1.

Varianta Pascal:

Listing OPER.PAS

```
Program Operatori;
var s:string[10];
    a:array[1..11] of Longint;
    val:Longint;
    k,n:Integer;
Begin
  Write('Introduceti operatorii:');
  Readln(s);
  n:=length(s);
  Write('Introduceti valoarea ',
                        'expresiei:');
  Readln(val);
  k:=2;
  a[1]:=val; { valoarea primului fac- }
             {  tor din primul termen }
  while s[k-1]='*' do
    begin a[k]:=1; Inc(k) end;
   { valoarea celorlalti factori din }
   {                  primul termen }
  while k<=n+1 do
    begin a[k]:=0; Inc(k) end;
            { valoarea factorilor din }
            {       restul termenilor }
  for k:=1 to n do
    Write(a[k],s[k]);
                { afisarea expresiei }
  Writeln(a[n+1],'=',val);
End.
```

Varianta C:

Listing OPER.CPP

```
#include <stdio.h>
#include <string.h>

void main(void){
  char s[10];
  long a[11],val,k=1,n;
  printf("Introduceti operatorii:");
  scanf("%s",&s);
  n=strlen(s);
  printf("Introduceti valoarea \
                        expresiei:");
  scanf("%ld",&val);
  a[0]=val; // valoarea primului fac-
            // tor din primul termen
  while (s[k-1]=='*') a[k++]=1;
        // valorile celorlalti factori
        // din primul termen
  while (k<=n) a[k++]=0;
        // valorile factorilor din
        // restul termenilor
  for (k=0;k<n;k++)
                  // afisarea expresiei
    printf("%ld%c",a[k],s[k]);
  printf("%ld=%ld\n",a[n],val);
}
```

Subiectul 4

Fișierul text **ELEVI.TXT** din directorul curent cuprinde pe fiecare din liniile sale numele, prenumele și mediile semestriale la informatică ale elevilor dintr-o clasă. Să se calculeze și să se scrie în fișierul **MEDII.TXT** media generală a clasei, știind că mediile corigenților nu se iau în considerare. Numele corigenților se vor scrie în fișierul **CORIG.TXT**.

Soluție

Varianta Pascal:

Listing ELEVI.PAS

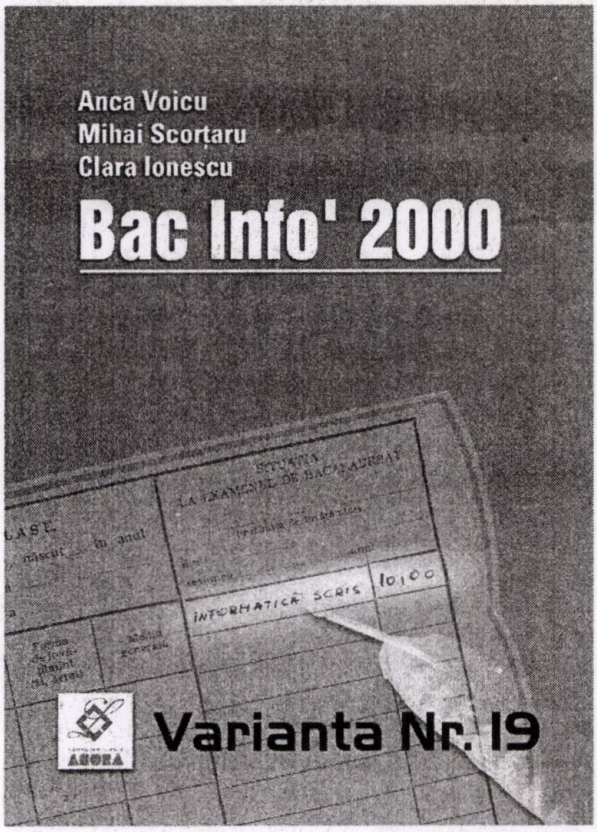
```
Program elevi;
type elev=record
           nume,pren:string[10];
           medie:Real
         end;
var f,g,h:Text;
    a:elev; i,k:Byte;
    car:Char;
    s,med_gen:Real;
Begin
  Assign(f,'ELEVI.TXT');
  Reset(f);
  Assign(g,'MEDII.TXT');
  Rewrite(g);
  Assign(h,'CORIG.TXT');
  Rewrite(h);
  while not Eof(f) do
  begin
    Read(f,car);
    with a do
    begin
      nume:='';
```

```
      while car<>' ' do
      begin
        nume:=nume+car;
        Read(f,car)
      end;
      pren:='';
      Read(f,car);
      while car<>' ' do
      begin
        pren:=pren+car;
        Read(f,car)
      end;
      Readln(f,medie);
      if medie>=5 then
        begin
          s:=s+medie;
          Inc(k)
        end
      else Writeln(h,nume,' ',pren)
    end
  end;
  med_gen:=s/k;
  Writeln(g,med_gen:10:2);
  Close(f);
  Close(g);
  Close(h)
End.
```

Varianta C:

Listing ELEVI.CPP

```
#include <stdio.h>
void main(void){
  struct elev{
    char nume[10],pren[10];
    float medie;
  } a;
  float s,med_gen;
  int k=0;
  FILE *f=fopen("ELEVI.TXT","rt"),
       *g=fopen("MEDII.TXT","wt"),
       *h=fopen("CORIG.TXT","wt");
  while (!feof(f)){
    fscanf(f,"%s%s%f",a.nume,a.pren,
                                a.medie);
    if (a.medie>=5) s+=a.medie,k++;
    else fprintf(h,"%s %s\n",
                        a.nume,a.pren);
  }
  med_gen=s/k;
  fprintf(g,"%10.2d\n",med_gen);
  fclose(f);
  fclose(g);
  fclose(h);
}
```



Barem

Subiectul 1: 30 puncte
1) 7,5 puncte
2) 7,5 puncte
3) 15 puncte
Subiectul 2: 20 puncte
1) 10 puncte
2) 10 puncte
Subiectul 3: 20 puncte
Subiectul 4: 20 puncte
1)10 puncte
2) 10 puncte
Se acordă 10 puncte din oficiu.
TOTAL: 100 puncte

Erată...

În numărul 2/2000 al Gazetei de Informatică a apărut o eroare în prezentarea soluției variantei nr. 17 din cartea Bac Info' 2000.

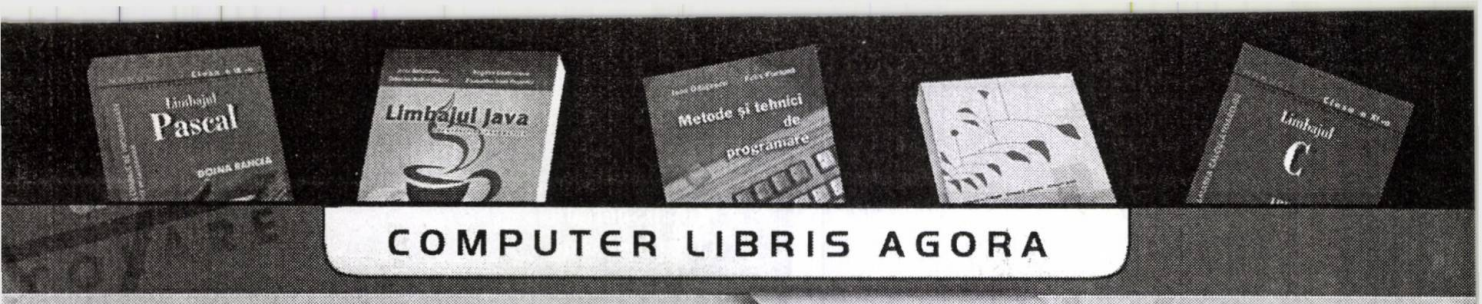
În descrierea algoritmului de la punctul 1 al primului subiect condiția din linia 9:

```
cât timp m ≠ n
```

trebuie înlocuită cu:

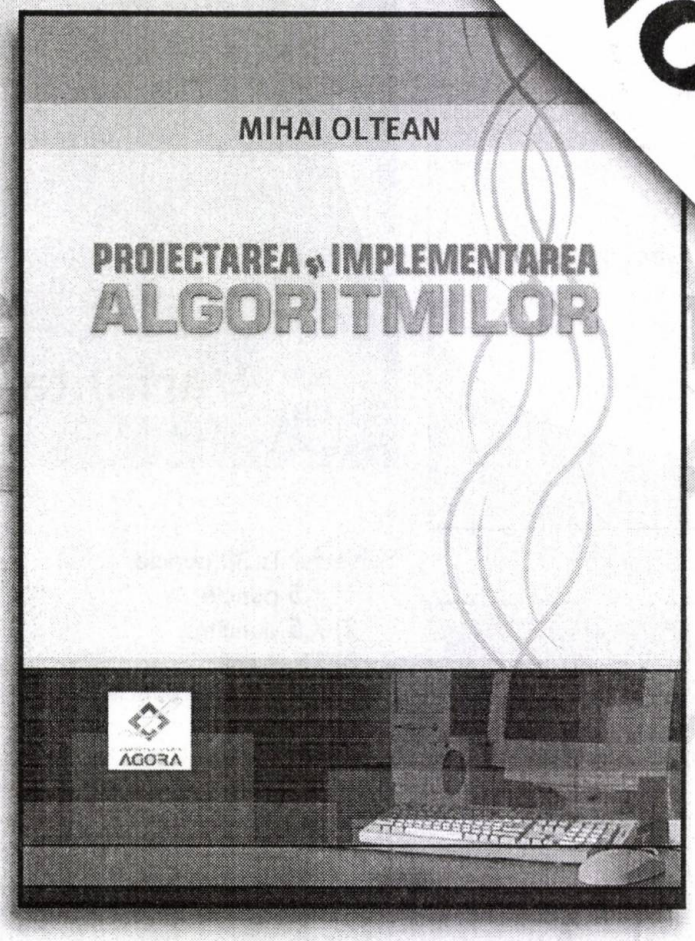
```
cât timp m > n
```

Redacția GInfo își cere scuze pentru această eroare!



COMPUTER LIBRIS AGORA

NOUTATE



Culegerea constituie un instrumentar pentru cei care doresc să intre în posesia secretelor proiectării algoritmilor. Va fi util când vă pregătiți pentru teză, pentru examenul de bacalaureat sau pentru admitere. Veți găsi în culegere și probleme care pot fi analizate și rezolvate în cercurile de elevi și care vor fi utile în pregătirea pentru concursurile școlare.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Instrucțiunea SELECT SQL

Ion Cozac

În articolul din numărul 5/99 am făcut cunoștință cu cîteva instrucțiuni ale limbajului SQL: Create table, Insert, Update, Delete. Aceste instrucțiuni ne-au permis să creăm tabele, să introducem și să actualizăm informații în aceste tabele. În articolul de față prezentăm câteva posibilități de interogare a bazelor de date relaționale cu ajutorul instrucțiunii Select SQL. Deoarece aproape fiecare SGBD relațional implementează un subset al limbajului SQL, plus unele "inovații" care creează uneori confuzii, insistăm doar asupra unor aspecte pe care le considerăm esențiale.

Limbajul SQL (*Structured Query Language*) a fost creat prin anii '70 în laboratoarele din *San Jose* ale firmei IBM. Sintaxa acestui limbaj este, după cum ne-am putut da seama, destul de apropiată de limba engleză, fiind adecvat programării pe calculator și ușor de asimilat de către utilizatori. SQL nu este un limbaj de programare (așa cum sunt limbajele procedurale cunoscute: *C*, *Pascal*, *Fortran*, *Cobol* etc.), este destinat manipulării datelor dintr-o bază de date relațională. De aceea el este inclus într-un limbaj procedural cu care este dotat aproape fiecare SGBD relațional. Sintaxa generală a instrucțiunii `Select SQL` este următoarea:

`SELECT [DISTINCT]` *atribut/expresie...* `FROM` *tabelă* `[AS sinonim]... [WHERE` *condiții de selecție și/sau condiții de legătură...*`] [GROUP BY` *atribut...*`] [HAVING` *condiții de selecție...*`] [ORDER BY` *atribut* `ASC|DESC...];`

Rezultatul selecției este o tabelă ale cărei coloane sunt câmpurile care apar în lista clauzei `Select`. Pentru a ilustra mai multe posibilități de utilizare a instrucțiunii `Select SQL`, vom lucra cu cinci exemple de baze de date. Numele tabelelor și ale cîmpurilor au fost alese în așa fel încât să sugereze cât mai bine semnificația lor, precum și să evidențieze cât mai clar legăturile dintre tabele. În articolul următor aceste exemple vor fi reluate și explicate, astfel că în final vom înțelege de ce o bază de date trebuie să fie compusă din așa de multe tabele. Pentru început trebuie să știm că în felul acesta se garantează corectitudinea datelor introduse (sau cel puțin există un control din partea SGBD-ului), se elimină redundanțele (repetițiile inutile), și eventuale actualizări se pot face ușor fără a afecta integritatea informațiilor deja introduse.

Selectarea unor informații dintr-o tabelă

Exemplul 1 - o evidență simplificată a studenților:

`STUDNOTE(Stud,Disc,Nota)`

Pentru a ilustra utilizarea instrucțiunii `Select` folosind o singură tabelă, vom face pentru început referire la tabela `StudNote` din primul exemplu care conține informații despre notele obținute de fiecare student la toate disciplinele pe care le studiază.

Structura tabelei este foarte simplă, avînd doar trei coloane (`Stud`, `Disc`, `Nota`), primele două formând cheia primară.

Cel mai simplu exemplu de instrucțiune `Select` este următorul:

```
Select * from StudNote;
```

Această instrucțiune selectează toate informațiile din tabela `StudNote`. Caracterul `'*'` se referă la toate coloanele tabelei (sau tabelelor - în cazul general) din lista clauzei `from`. Putem selecta o parte din coloanele tabelei, ca în exemplul de mai jos:

```
Select Disc from StudNote;
```

Această instrucțiune selectează lista disciplinelor care sunt studiate. Dacă o folosim în forma prezentată mai sus, vom observa că tabela rezultat va avea tot atâtea linii câte are și tabela `StudNote`, aceasta deoarece fiecare disciplină apare de mai multe ori (o dată pentru fiecare student).

Următorul exemplu arată cum se poate obține lista studenților, de data aceasta fără repetiții supărătoare:

```
Select distinct Stud from StudNote;
```

Putem cere selectarea unor linii care îndeplinesc anumite condiții, de exemplu lista notelor sub 5:

```
Select * from StudNote where Nota<5;
```

În clauza `where` putem preciza condiții complexe de căutare, combinate cu operatorii logici `AND`, `OR` și `NOT`, ordinea de evaluare fiind cea cunoscută. Se pot folosi și paranteze pentru a controla ordinea de evaluare a condițiilor.

Funcții de agregare în interogări cu o singură tabelă

Dacă ne interesează care este cea mai mare și cea mai mică notă dintre cele care au fost date, formulăm următoarea interogare:

```
Select min(Nota), max(Nota) from StudNote;
```

Media generală se obține astfel:

```
Select avg(Nota) from StudNote;
```

Folosim funcția count pentru a afla numărul de înregistrări ale tabelei StudNote:

```
Select count(*) from StudNote;
```

Standardul SQL definește următoarele funcții de agregare:

- min: valoarea minimă de pe o anumită coloană;
- max: valoarea maximă de pe o anumită coloană;
- sum: suma valorilor de pe o anumită coloană;
- avg: media aritmetică a valorilor de pe o anumită coloană;
- count: numărul de înregistrări selectate.

Funcțiile sum și avg pot fi folosite doar pentru coloane care au valori numerice, spre deosebire de celelalte care pot opera cu orice tip de valori. Funcția avg poate returna o valoare de tip diferit de cel al atributului căruia i se aplică (de exemplu, media aritmetică a unor valori întregi poate fi o valoare cu zecimale). Sunt necesare câteva precizări în legătură cu aceste funcții. Implicit ele iau în considerare toate valorile aflate pe coloana specificată; dacă se dorește să fie prelucrate doar valorile distincte, atunci numele coloanei trebuie precedat de cuvîntul cheie distinct.

Precizăm că funcția count poate fi folosită pentru a număra toate înregistrările - forma count(*), sau pentru a număra valorile distincte ale unei anumite coloane. De exemplu, ne interesează câți studenți avem în evidență:

```
Select count(distinct Stud) from StudNote;
```

Să presupunem că ne interesează notele cele mai mari, dar și care sunt studenții care au primit acele note. Pentru aceasta trebuie să comparăm fiecare notă cu valoarea maximă, care poate fi obținută printr-o interogare. Deducem că trebuie să folosim interogări SQL imbricate:

```
Select * from StudNote where Nota =
(Select max(Nota) from StudNote);
```

Comparația este posibilă deoarece instrucțiunea Select interioară returnează o singură valoare.

Acestea au fost interogări relativ simple. Dacă dorim să știm care studenți sunt restanțieri, interogarea este de asemenea foarte simplă:

```
Select Distinct Stud from StudNote where Nota<5;
```

Am folosit clauza distinct deoarece, altfel, studenții care au mai mult de o restanță apar de mai multe ori în listă.

Dacă în schimb dorim să știm care sunt studenții integraliști, interogarea ne va da serioase dureri de cap. Aceasta deoarece, spre deosebire de cazul precedent, aici trebuie să formulăm altfel condițiile de selecție. Prezentăm în continuare două formulări echivalente:

- toate notele obținute de un student sunt de cel puțin 5;
- nici o notă obținută de un student nu este sub 5.

Ar trebui ca pentru fiecare student să încercăm să selectăm numai înregistrările corespunzătoare acestuia și care au nota sub 5. Se reține în rezultatul final numai acei studenți pentru care nu s-a selectat nici o înregistrare (rezultatul selecției este vid). În pseudocod această problemă se rezolvă astfel:

```
pentru i=1,n
   S ← ∅;
   pentru j=1,n
     dacă (StudNote[i].Stud = StudNote[j].Stud
                    și StudNote[j].Nota < 5)
       S ← S ∩ StudNote[j];
dacă (S = ∅)
  output(StudNote[i].Stud);
```

În secvența de mai sus am comparat câmpul Stud din înregistrările StudNote[i] și StudNote[j]. În limbajul SQL nu avem o astfel de posibilitate; putem în schimb să folosim o tabelă sub mai multe nume (sinonime), și să comparăm două câmpuri ca și cum ele ar face parte din tabele diferite. Interogarea se rezolvă astfel:

```
Select Stud from StudNote as S where not exists
(Select * from StudNote as T
where S.Stud = T.Stud and T.Nota < 5);
```

Remarcăm folosirea expresiei condiționale not exists, care returnează valoarea true dacă tabela transmisă ca argument este vidă.

Până acum, funcțiile de agregare au operat asupra tuturor valorilor din rezultatul selecției. Putem să construim și grupe de înregistrări asupra cărora să opereze aceste funcții. Ne interesează media fiecărui student:

```
Select Stud, avg(Nota) from StudNote
group by Stud;
```

Putem ordona această situație descrescător după medie și în ordine alfabetică după nume:

```
Select Stud, avg(Nota) from StudNote
group by Stud order by 2 desc, 1;
```

Dacă dorim să selectăm doar acei studenți care au media minimum 9, folosim și clauza having:

```
Select Stud, avg(Nota) from StudNote
group by Stud having avg(Nota) >= 9;
```

Selectarea unor informații din mai multe tabele

Exemplul 2 - evidența angajaților unei instituții:

```
DEPARTAMENTE(Codd,Dendep,Sefd)
PERSONAL(Nrleg,Nume,Dep)
```

Ne interesează să aflăm, pentru fiecare departament, cine este șeful acestuia și câți angajați lucrează la departamentul respectiv:

```
Select Codd, Dendep, Sefd, Nume,
       (Select count(*) from Personal
        where Codd = Dep)
from Departamente,Personal where Sefd = Nrl;
```

Remarcăm și în acest caz folosirea unei instrucțiuni Select care returnează o valoare. Pentru această instrucțiune Select atributul Codd joacă același rol pe care îl are o variabilă globală într-un program C sau *Pascal*. Aceasta deoarece ne interesează câți angajați are un anumit departament: dacă am fi inclus în lista clauzei from și tabela Departamente, atunci instrucțiunea Select interioară ar fi pierdut valoarea atributului Codd din exterior (să ne gândim ce se întâmplă atunci când scriem un program în

C sau *Pascal*). Dar lucrurile nu s-ar opri aici: instrucțiunea Select interioară ar efectua produsul cartezian al celor două tabele și ar număra câte înregistrări ale acestui produs îndeplinesc condiția din clauza where.

În instrucțiunea Select exterioară se remarcă în clauza where o condiție care se numește *condiție de legătură*. Aceste condiții de legătură se scriu în general astfel:

```
cheie-primară = cheie-externă
```

Exemplul 3 - evidența facturilor:

```
CLIENTI(Codcl,Nume,Localit,Adresa)
PRODUSE(Codpr,Denpr,Pretu)
FACTURI(Nrf,Dataf,Codcl)
DETALII(Nrf,Codpr,Cant)
```

Dorim să știm care este valoarea fiecărei facturi:

```
Select Facturi.Nrf, Dataf, Facturi.Codf,
Nume, Localit, Adresa,
       Sum(Cant*Pretu)
from Facturi, Clienti, Detalii, Produse
where Facturi.Nrf = Detalii.Nrf
and Facturi.Codcl = Clienti.Codcl
and Detalii.Codpr = Produse.Codpr
group by Facturi.Nrf, Dataf, Facturi.Codc,
Nume, Localit, Adresa;
```

Remarcăm mai întâi necesitatea de a califica acele atribute care apar cu același nume în mai multe tabele. În al doilea rând (un fapt nu tocmai plăcut), necesitatea de a include în lista clauzei group by toate câmpurile din clauza Select care nu sunt funcții de agregare. Standardul SQL cere acest lucru aparent lipsit de logică și majoritatea SGBD-urilor care implementează limbajul Select SQL se conformează acestei cerințe. În al treilea rând observăm că în lista clauzei Select putem avea și expresii aritmetice, chiar și ca argumente ale unei funcții de agregare.

O altă interogare pe care o putem face este: suma totală pe care a plătit-o fiecare client. Aceasta este ceva mai scurtă decât precedenta:

```
Select Facturi.Codcl, Nume, Localit, Adresa,
Sum(Cant*Pretu)
from Facturi, Clienti, Detalii, Produse
where Facturi.Nrf = Detalii.Nrf
and Facturi.Codcl = Clienti.Codcl
and Detalii.Codpr = Produse.Codpr
group by Facturi.Codf, Nume, Localit, Adresa;
```

Dacă dorim să aflăm căror clienți le-am întocmit facturi care totalizează o valoare maximă, trebuie să procedăm în doi pași. Mai întâi creăm o tabelă nouă în care depunem rezultatele selecției de mai sus. Pentru aceasta folosim instrucțiunea Insert cu următoarea sintaxă, presupunând că avem creată o tabelă Valori (Codf,Nume, Localit,Adresa,Val):

```
Insert into Valori
Select Facturi.Codf, Nume, Localit, Adresa,
Sum(Cant*Pretu)
from Facturi, Clienti, Detalii, Produse
where Facturi.Nrf = Detalii.Nrf
and Facturi.Codcl = Clienti.Codcl
and Detalii.Codpr = Produse.Codpr
group by Facturi.Codf, Nume, Localit, Adresa;
```

La pasul al doilea putem extrage informațiile care ne interesează:

```
Select Codcl, Nume, Localit, Adresa, Val
from Valori where Val =
(Select max(Val) from Valori);
```

Exemplul 4 - evidența comenzilor

```
FURNIZORI(Codf,Nume,Localit,Adresa)
PRODUSE(Codp,Denp)
OFERTE(Codf,Codp,Pretu)
COMENZI(Nrc,Datac,Codf)
DETALII(Nrf,Codp,Cant)
```

Dorim să știm care este valoarea fiecărei comenzi:

```
Select Comenzi.Nrc, Datac, Comenzi.Codf,
Nume, Localit, Adresa,
       Sum(Cant*Pretu)
from Furnizori, Oferte, Comenzi, Detalii
where Comenzi.Nrc = Detalii.Nrf
and Detalii.Codp = Oferte.Codp
and Comenzi.Codf = Furnizori.Codf
and Detalii.Codp = Oferte.Codp
and Comenzi.Nrc = Oferte.Codf
group by Comenzi.Nrc, Datac, Comenzi.Codf,
Nume, Localit, Adresa;
```

Am fost nevoiți să punem toate cele patru condiții de legătură, deoarece informațiile despre prețurile produselor se află în tabela Oferte - fiecare produs poate avea prețuri diferite de la un furnizor la altul.

Ne interesează de unde putem achiziționa fiecare produs la cel mai mic preț:

```
Select Produse.Codp, Denp, Pretu,
Furnizori.Codf, Nume, Localit, Adresa
from Oferte as O, Furnizori as F, Produse as P
where O.Codf =
(Select Codf from Oferte as Q
 where P.Codp = Q.Codp and Pretu =
 (Select min(Pretu) from Oferte))
and O.Codf = F.Codf and O.Codp = P.Codp;
```

Exemplul 5 - o evidență detaliată a studenților:

```
STUDENTI(Nrls,Nume,Spec,An)
DISCIPLINE(Codd,Dend,Spec,An)
SPECIALIZARI(Cods,Dens)
NOTELE(Nrls,Codd,Nota)
```

Această evidență presupune gestiunea a patru tabele. Primele trei tabele le numim tabele primare, deoarece datele primare se introduc în tabelele Studenti, Discipline, Specializari. Tabela Notele păstrează informații care presupun o corelare între celelalte trei tabele, deci nu putem introduce oricum datele de intrare. Presupunem că avem toate datele introduse în tabelele primare și dorim să pregătim tabela Notele pentru a putea introduce ulterior

și notele de la examene. Folosim o instrucțiune `Insert` pentru a introduce în tabelă rezultatul unei selecții:

```
Insert into Notele (Nrls, Codd)
Select Nrls, Codd from Studenti, Discipline
where Studenti.Spec = Discipline.Spec
and Studenti.An = Discipline.An;
```

Instrucțiunea `Select` de mai sus efectuează produsul cartezian al celor două tabele, și selectează acele înregistrări pentru care sunt îndeplinite condițiile din clauza `where`. Aceste condiții nu sunt de legătură, deoarece între atributele implicate nu există o legătură de tip cheie primară - cheie externă.

După ce am introdus notele dorim o situație a mediilor, ordonată pe specializări, ani de studiu, medie (descrescător), nume (alfabetic):

```
Select Notele.Nrls, Nume, Spec, Dens, An,
avg(Nota) as Media
from Notele, Studenti, Specializari
where Notele.Nrls = Studenti.Nrls
and Studenti.Spec = Cods
group by Notele.Nrls, Nume, Spec, Dens, An
order by 3,5,6,2;
```

Utilizarea instrucțiunii Select SQL în actualizare

Am văzut mai sus cum se utilizează instrucțiunea `Select SQL` în conexiune cu instrucțiunea `Insert`. Putem cere în plus actualizarea (modificarea sau ștergerea) unor înregistrări care îndeplinesc anumite condiții, și care se exprimă cu ajutorul unor selecții.

Un exemplu: în preajma unor sărbători (de Crăciun sau Paște) un furnizor (căruia îi știm numele, dar nu și codul) anunță reduceri de preț cu 15%. Aceste modificări se operează astfel:

```
UPDATE Oferte SET Pretu = Pretu * 0.85
  WHERE Codf =
  (SELECT Codf FROM Furnizori
    WHERE Nume = "...");
```

În locul punctelor de suspensie introducem numele furnizorului. În mod asemănător se poate folosi instrucțiunea `Select SQL` în conexiune cu instrucțiunea `Delete`.

Ion Cozac este lector la Universitatea "Petru Maior" din Târgu-Mureș și poate fi contactat prin e-mail la adresa cozac@uttgm.ro.

news...

Parteneriat între Oracle și TurboLinux

TurboLinux a anunțat lărgirea relațiilor de colaborare ale companiei cu Oracle, aceasta din urmă achiziționând și o parte din acțiunile primei companii.

În schimb, TurboLinux va optimiza sistemul său de operare pentru Oracle 8i. Optimizarea și certificarea sistemului de operare TurboLinux Server pentru Oracle 8i este pasul final al companiei pentru furnizarea de soluții Linux care să suporte aplicații e-business de mare volum.

Pentru a face față cererii tot mai mari de soluții tip enterprise pentru sistemul de operare Linux de pe piața asiatică, Oracle, TurboLinux și Huadi, cel mai mare integrator de sisteme din China, plănuiesc să colaboreze pentru a oferi astfel de soluții clienților guvernamentali și marilor companii.

Sistemul de operare Linux câștigă din nou teren față de Windows, tot mai multe companii preferând să utilizeze soluții Linux în desfășurarea acticvității.

Venituri record la Sun Microsystems

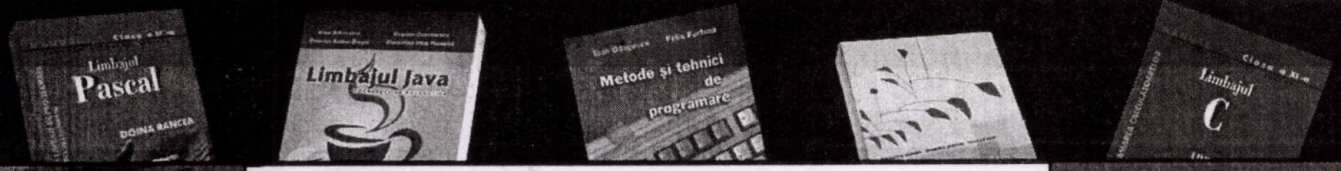
Compania Sun Microsystems, unul dintre liderii mondiali în domeniul IT, a trecut pentru prima dată în istoria sa de pragul de 4 miliarde de dolari încasați într-un singur trimestru, cel puțin așa reiese din raportul financiar referitor la trimestrul încheiat în 26 martie 2000, pus la dispoziția presei.

Încasările pentru acest trimestru au crescut cu 35% față de aceeași perioadă a anului trecut.

Câștigurile nete ale companiei s-au ridicat, tot în această perioadă, la suma de 436,2 de milioane de dolari, cu 49% mai mult față de aceeași perioadă a anului trecut.

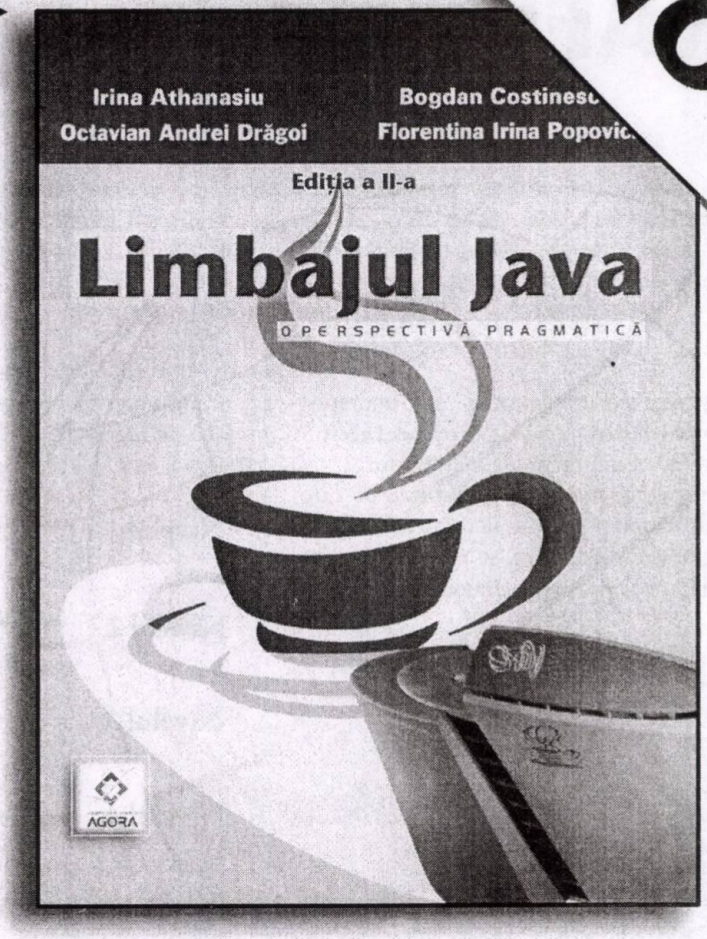
Pentru primele trei trimestre ale anului fiscal 2000, Sun a raportat încasări în valoare de 10,704 miliarde de dolari, cu 29% mai mult decât în aceeași perioadă a anului trecut.

Câștigurile nete au atins cota de 1,065 miliarde de dolari, marcând o creștere cu aproximativ 40% față de aceeași perioadă a anului 1999.



COMPUTER LIBRIS AGORA

NOUTATE



De ce Java ?
Deoarece tot ce se întâmplă acum în lumea calculatoarelor este pentru Internet, legat de Internet sau accesibil din Internet. Java este limbajul de programare care a devenit un standard de-facto în toate aceste cazuri. Java este acum mai mult decât un limbaj de programare, este o colecție de tehnologii pentru Internet.
De ce reeditată ?
Datorită dinamicii evoluției limbajului, autorii au considerat necesară verificarea exemplelor și pe versiunea Java 2 precum și adăugarea a două noi capitole: componente Java Beans și securitate.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Introducere în teoria GRAFURILOR

Cristian Cadar

În cadrul acestui episod ne propunem să discutăm câteva aspecte legate de existența ciclurilor într-un graf: verificarea ciclicității grafurilor, determinarea ciclurilor elementare, găsirea muchiilor și a punctelor critice într-un graf, testarea biconexității și găsirea componentelor biconexe.

Cicluri

Determinarea existenței ciclurilor într-un graf, împreună cu toate celelalte probleme enunțate în prezentarea acestui articol, sunt aplicații ale parcurgerii DF a grafurilor.

Presupunem în cele ce urmează că grafurile pe care vom exemplifica algoritmii prezentați în acest articol sunt *conexe*. În cazul grafurilor neconexe, se vor aplica acești algoritmi pentru fiecare componentă conexă.

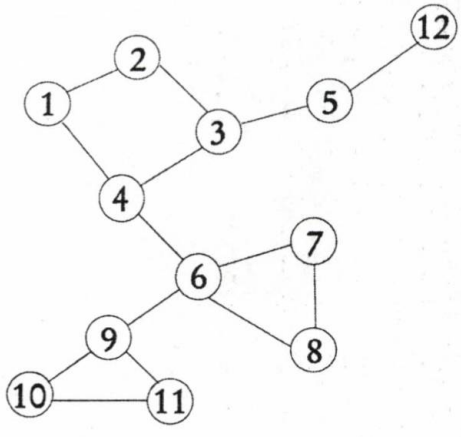


Figura 1

Problema determinării existenței ciclurilor într-un graf conex este o problemă deosebit de simplă. În primul rând, să ne reamintim că un arbore parțial al unui graf este conex și fără cicluri. De asemenea, acesta conține n-1 muchii. Am arătat într-un articol anterior că un arbore parțial este maximal în raport cu aciclicitatea, adăugarea oricărei muchii în plus determinând apariția ciclurilor. Prin urmare, pentru a determina existența ciclurilor într-un graf neorientat conex, trebuie doar să testăm dacă graful are n-1 muchii. Metoda care este însă cea mai folosită pentru rezolvarea acestei probleme este, așa cum am precizat, parcurgerea DF. Pentru aceasta, vom prezenta conceptul de *"etajare a grafului pe niveluri"*. Să considerăm graful din figura 1. Etajarea acestuia pe niveluri este reprezentată în figura 2.

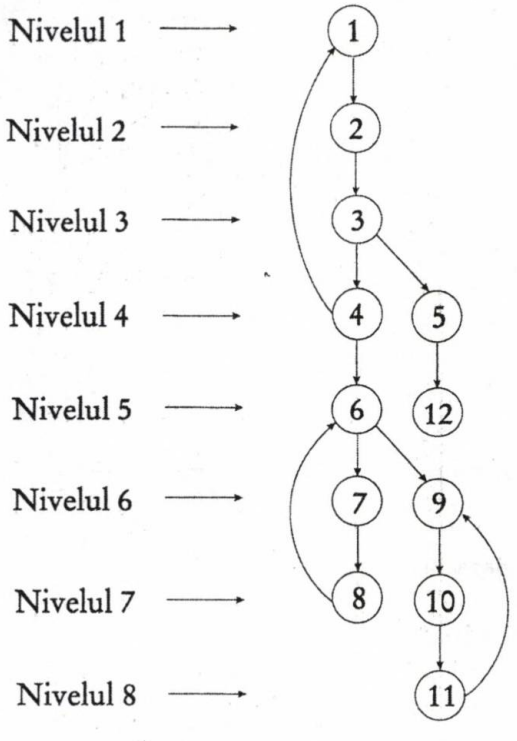


Figura 2: Etajarea pe niveluri a unui graf

Împărțirea grafurilor pe niveluri se face printr-o parcurgere DF. Nivelul unui nod reprezintă adâncimea la care a fost vizitat nodul respectiv în parcurgerea DF. De exemplu, nodurile 7 și 9 se află pe nivelul 6, nodul 11 pe

nivelul 8, iar nodul 1 pe nivelul 1. Atașarea nivelelor corespunzătoare nodurilor grafului, se face ușor printr-o parcurgere DF:

```
Parcurge_DF(nod k, level L)
Vizitat(k) ← Adevărat
Nivel(k) ← L
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
  dacă nu Vizitat(i)
    atunci Parcurge_DF(i, L+1)
  i ← următorul_vecin(k)
```

În ceea ce privește muchiile, etajarea pe niveluri a grafului le împarte în două categorii: *muchii de avansare* și *muchii de întoarcere*. Muchiile de avansare sunt muchiile care leagă un nod de pe un nivel inferior cu unul de pe un nivel imediat superior, deci muchiile care realizează avansarea de la un nivel la altul al grafului. În exemplul considerat acestea sunt (1, 2), (2, 3), (3, 4), (3, 5), (4, 6), (5, 12), (6, 7), (6, 9), (7, 8), (9, 10), (10, 11).

Referitor la muchiile de întoarcere acestea sunt muchiile care unesc un nod aflat pe un nivel superior cu unul aflat pe un nivel inferior, diferența absolută dintre nivelurile acestor noduri fiind cel puțin 2. Pentru graful din figură, muchiile de întoarcere sunt: (4, 1), (8, 6) și (11, 9).

Bineînțeles, muchiile care nu sunt de avansare sunt muchii de întoarcere și invers. De asemenea, trebuie reținut faptul că, privite ca muchii de avansare și de întoarcere, muchiile grafului sunt *orientate*, deci sunt considerate arce.

Revenind la problema determinării existenței ciclurilor într-un graf, remarcăm faptul că muchiile de avansare determină un arbore parțial al grafului respectiv. Prin urmare, existența ciclurilor într-un graf neorientat devine echivalentă cu existența a cel puțin unei muchii de întoarcere în acel graf. Acest test se face ușor, complexitatea algoritmului fiind $O(n+m)$:

```
Există_Ciclu(nod k)
Vizitat(k) ← Adevărat
i ← primul_vecin(k)
cât timp nu am parcurs toată lista vecinilor lui k
                    // considerăm toți vecinii lui k mai
                    // puțin nodul din care a fost apelat
  dacă nu Vizitat(i)
    atunci Există_Ciclu(i)
    altfel scrie "Am găsit ciclu !"
  i ← următorul_vecin(k)
```

De multe ori suntem interesați să găsim numărul maxim de *cicluri disjuncte între ele*, adică numărul maxim de cicluri cu proprietatea că oricare dintre ele conține cel puțin o muchie care nu apare în nici un alt ciclu, muchie pe care o vom numi *muchie proprie* a ciclului respectiv. De exemplu, pentru graful din figura 3, două astfel de partiționări în cicluri sunt reprezentate în figurile 4 și respectiv 5:

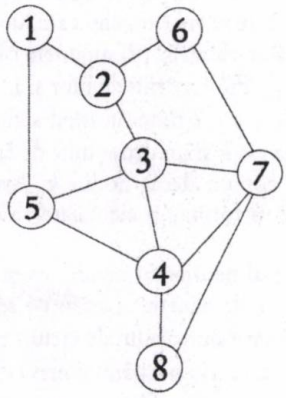


Figura 3

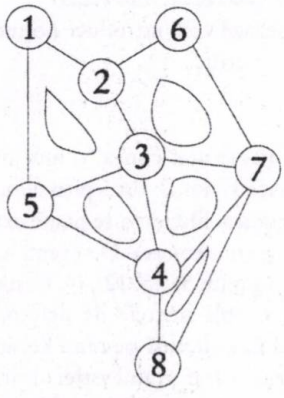


Figura 4: O posibilitate de partiționare în cicluri elementare

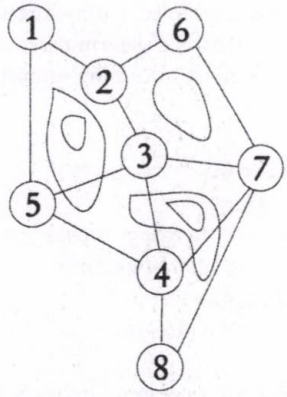


Figura 5: O altă posibilitate de partiționare în cicluri elementare

Mulțimea de cicluri care îndeplinesc această proprietate se numește mulțimea *ciclurilor elementare*. Remarcăm că într-un graf conex neorientat, numărul de cicluri care îndeplinesc această proprietate poate fi cel mult $m-(n-1)$, deoarece un arbore parțial (cu $n-1$ muchii) este graful parțial cel mai mic (din punctul de vedere al numărului de muchii) care nu conține cicluri. În cele ce urmează vom arăta constructiv că pentru orice graf neorientat conex, putem construi $m-n+1$ cicluri. Pentru aceasta vom considera un arbore parțial oarecare al grafului dat. Pentru sim-

plitate, vom considera chiar arborele parțial obținut în urma parcurgerii DF. Se observă ușor că prin introducerea fiecăreia dintre cele m-n+1 muchii care nu aparțin arborelui, se formează un ciclu. Să presupunem că avem de introdus muchia (i, j). Fie k părintele lui i și j, adică nodul cel mai apropiat de i și j. Atunci în mod sigur, vom avea un drum unic de la i la k și un drum unic de la j la k, drumuri care nu au în comun decât nodul k. Atunci, adăugând muchia (i, j) se va forma un ciclu care va conține muchia proprie (i, j).

Pentru un graf neorientat conex, avem deci un număr de m-n+1 cicluri elementare. Lăsăm pe seama cititorului sarcina determinării numărului de cicluri elementare pentru un graf oarecare (nu neapărat conex) cât și implementarea algoritmului pentru aflarea acestor cicluri.

Muchii și puncte critice

În cele ce urmează vom introduce noțiunea de muchie critică și de punct critic.

Definiție

Într-un graf neorientat conex, o muchie se numește *muchie critică* dacă și numai dacă prin eliminarea ei graful își pierde conexitatea. Definiția se poate extinde ușor și la nivel de componente conexe. De exemplu, în figura 1, muchiile critice sunt (3, 5), (5, 12), (4, 6) și (6, 9).

Cea mai simplă metodă de determinare a muchiilor critice constă în a elimina pe rând fiecare muchie a grafului și de a verifica dacă graful astfel obținut mai este conex. Dacă graful astfel obținut nu mai este conex, atunci, conform definiției, muchia este critică, în caz contrar ea este o muchie oarecare a grafului. Complexitatea acestui algoritm este $O(m^2)$, deoarece determinarea conexității unui graf se face așa cum am arătat într-un articol anterior în $O(n+m)$:

```
boolean Critice(muchie (x,y))
elimină muchia (x,y)
dacă Conex(G)  // funcție prezentată în cadrul unui
               // episod anterior
  atunci Critice ← Fals
  altfel Critice ← Adevărat
```

Pentru a prezenta algoritmul optim de găsire a muchiilor critice într-un graf, algoritm de complexitate $O(n+m)$, observăm mai întâi că muchiile critice sunt muchiile care nu aparțin nici unui ciclu. Pentru a le determina, vom folosi etajarea grafului pe niveluri. În primul rând remarcăm că o muchie de întoarcere nu poate fi critică, deoarece o astfel de muchie închide întotdeauna un ciclu. Rămâne de verificat care dintre muchiile de avansare (care sunt în număr de n-1) fac parte dintr-un ciclu. Să considerăm că dorim să verificăm muchia de avansare de la i la j. Afirmăm că, dacă din nodul j putem să ajungem pe un nivel mai mic sau egal cu nivelul lui i (avansând în sensul muchiilor de la parcurgerea DF), atunci muchia nu este critică, în caz contrar ea este critică. Demonstrația acestei afirmații este ușor de intuit: în momentul în care ajungem pe un nivel mai mic sau egal cu nivelul lui i, închidem un ciclu, deci muchia nu este critică. În caz contrar, nu există nici un ciclu care să conțină muchia (i, j), deci aceasta este critică. În exemplul de față, muchiile (4, 6), (6, 9), (3, 5) și (5, 12) sunt critice, celelalte fiind muchii normale. De exemplu, muchia (2, 3) nu este critică deoarece din 3 se poate ajunge pe drumul 3-4-1 în nodul 1, care se află pe nivelul 1, nivel mai mic decât nivelul 2 pe care se află nodul 2.

Vă prezentăm în continuare procedura de determinare a muchiilor critice în complexitatea $O(n+m)$ (parcurgere DF modificată), procedură care implementează algoritmul descris mai sus:

```
Muchii_Critice(nod k, level L, val_min)
                  // procedura recursivă apelată prin
                  // Muchii_Critice(1,1,aux)
dacă Vizitat(k)   // am ajuns într-un nod vizitat
  atunci val_min ← Nivel(k)
         * ieșire forțată din procedura recursivă
Vizitat(k) ← Adevărat;
Nivel(k) ← L;
val_min ← L; // inițializăm cu k valoarea minimă a
             // nivelului pe care poate ajunge nodul k
i ← primul_vecin(k) // considerăm toți vecinii lui k
                    // mai puțin nodul din care a fost apelat
cât timp nu am parcurs toata lista vecinilor lui k
  Muchii_Critice(i,L+1,aux)
  dacă aux<val_min
    atunci val_min ← aux
  dacă aux=L+1
    atunci scrie "Muchia (k,i) este critică"
  i ← următorul_vecin(k)
```

Definiție

Într-un graf neorientat conex, un nod se numește *punct critic* dacă și numai dacă prin eliminarea lui și a muchiilor adiacente acestuia, graful își pierde conexitatea. Ca și în cazul muchiilor critice, definiția poate fi extinsă și la nivel de componente conexe.

Algoritmii descriși în cazul aflării muchiilor critice pot fi folosiți cu mici modificări și pentru a afla punctele critice ale unui graf.

Primul algoritm presupune eliminarea pe rând a fiecărui nod împreună cu muchiile adiacente lui, urmată de verificarea conexității grafului astfel obținut. Algoritmul are complexitatea $O(m^2)$.

Cel de-al doilea algoritm, bazat pe etajarea grafului pe niveluri, este și el asemănător celui prezentat mai sus, cu specificarea faptului că, pentru ca un nod k să nu fie punct critic, trebuie ca pentru fiecare muchie de avansare (k, j), nodul j să poată ajunge pe un nivel mai mic decât nivelul nodului k. De remarcat faptul că nodul din care se începe parcurgerea (în general nodul 1) face excepție de la această regulă, deoarece nivelul său nu va putea fi depășit nicio-

dată, neexistând noduri pe niveluri mai mici decât 1. De aceea nodul 1 va fi considerat punct critic dacă și numai dacă are mai mulți fii. Numai în acest caz, nodul 1 va fi critic, el deconectând subarborii respectivi. Implementarea acestor algoritmi îi lăsăm pe seama cititorului.

Biconexitate

Definiție

Prin *graf biconex* înțelegem un graf conex cu proprietatea că eliminând oricare nod al acestuia, graful rămâne conex. Ca și în cazul conexității și tare conexității, definim *componentele biconexe* ca mulțimi maximale de noduri ale unui graf care respectă proprietatea de mai sus. În cele ce urmează, vom prezenta un algoritm eficient de determinare a componentelor biconexe ale unui graf.

De exemplu, pentru graful din figura 6, componentele biconexe sunt cele reprezentate în figura 7:

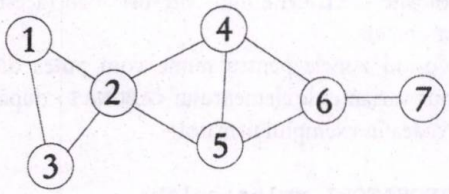


Figura 6

Remarcăm că împărțirea în componente biconexe nu introduce ca și până acum (conexitate, tare conexitate) o partiție a nodurilor grafului, ci *o partiție a muchiilor* acestuia.

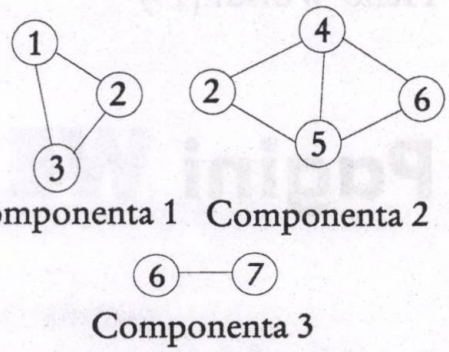


Figura 7: Componentele biconexe ale grafului din figura 6

De asemenea, se remarcă ușor că o componentă biconexă este o componentă care nu conține puncte critice. Astfel, pentru a determina componentele biconexe ale unui graf, vom adapta algoritmul de aflare a punctelor critice prezentat mai sus, reținând o stivă în care vom memora toate muchiile de avansare și de întoarcere parcurse până la un moment dat. Atunci când găsim un nod critic, se va forma o nouă componentă biconexă, pe care o vom determina extrăgând din stivă muchiile corespunzătoare. Complexitatea algoritmului este deci $O(n+m)$, iar implementarea acestuia o lăsăm pe seama cititorului, fiind foarte asemănătoare rutinei de determinare a muchiilor critice, a cărei descriere în pseudocod a fost prezentată în acest episod.

Cristian Cadar este student în anul II la Universitatea Politehnică din București. Poate fi contactat la adresa ccadar@pcnet.pcnet.ro.

news...

Star Office 5.2

Sun Microsystems, în eforturile sale de a lupta cu Microsoft Office a anunțat că versiunea Star Office 5.2 este disponibilă pentru download gratuit.

Aceasta versiune se concentrează pe îmbunătățirea suportului pentru formatele de fișiere ale produselor Microsoft pentru a ușura trecerea la StarOffice. În acest scop a fost îmbunătățit suportul pentru exportul și importul de fișiere în și din Microsoft Office.

Deși au fost adăugate peste 200 de noi facilități, Star Office nu se poate compara cu pachetul de programe oferit de Microsoft.

Produse Adobe

La Internet World, în Los Angeles, Adobe a anunțat noile produse GoLive 5.0 și Illustrator 9.0.

GoLive, o aplicație pentru crearea paginilor Web, ușurează lucrul cu codul HTML, oferă un editor multimedia îmbunătățit, suport complet pentru protocoalele W3C, precum și o integrare mai bună cu uneltele Adobe. Noua versiune va fi lansată în cursul lunii iunie.

Illustrator 9 dispune de suport Flash și SVG, pentru crearea automată de umbre și strălucire, precum și facilități de creare a transparențelor. Produsul va fi lansat tot în iunie.

Pagini WEB cu XML 1.0

Mihai Scorțaru

În cadrul acestui episod vom prezenta câteva detalii referitoare la zonele pentru nume folosite pentru a putea diferenția două elemente XML diferite care, din anumite motive au aceeași denumire. De asemenea, veți afla ce este modelul document obiect și cum poate fi folosit acesta pentru a descrie datele stocate în format XML.

Zone pentru nume

Una dintre cele mai importante probleme care poate apărea dacă se lucrează cu date în format XML este aceea că elementele pot avea aproape orice nume dorim. Din nefericire, aceasta înseamnă că există o probabilitate destul de mare ca, la un moment dat, să alegeți un nume pe care l-a mai ales cineva sau să folosiți același nume cu semnificații diferite în documente XML distincte.

Să presupunem că fișierul **COLAB.XML** conține elementul <**ABONAT**>Yes</**ABONAT**>, iar fișierul **ABONATI.XML** conține elementul <**ABONAT**>Tudor Oroian </ABONAT>. Aceasta nu ar fi o problemă dacă cele două documente rămân separate, dar dacă este nevoie să creăm un nou document care să conțină datele din aceste documente atunci ar putea apărea construcții de genul:

```
<COLABORATORI>
  <COLABORATOR>
    <ID>1001</ID>
    <ABONAT>Yes</ABONAT>
    <ABONAT>Oroian Tudor</ABONAT>
  </COLABORATOR>
</COLABORATORI>
```

În această situație nu vom mai putea ști care este semnificația elementului <**ABONAT**>.

Pentru a evita astfel de situații au fost introduse ***zonele pentru nume*** (engl. *namespace*) care au rolul de a identifica în mod unic schema căreia îi aparține un anumit element.

Zonele pentru nume pot fi adăugate unui document XML folosind atributul **xmlns** (*XML Name Space*) a cărui valoare este un URI (*Uniform Resource Identifier — Identificator uniform de resurse*).

Acest URI este doar un șir de caractere care poate identifica zone pentru nume. Chiar dacă puteți folosi orice secvență unică pentru acest URI, de cele mai multe ori sunt folosite URL-urile unor site-uri Web (acestea sunt evident unice).

Folosind zonele pentru nume vom putea diferenția cele două variante ale elementului <**ABONAT**> după cum se poate vedea în exemplul următor:

```
<COLABORATORI xmlns:colab=
                "http://www.ginfo.ro/colab"
  xmlns:abonati=
            "http://www.ginfo.ro/abonati">
  <COLABORATOR>
    <ID>1001</ID>
    <colab:ABONAT>Yes</colab:ABONAT>
    <abonati:ABONAT>Oroian Tudor
    </abonati:ABONAT>
  </COLABORATOR>
</COLABORATORI>
```

Am adăugat atribute **xmlns** pentru a identifica două zone pentru nume. Sintaxa pentru folosirea acestui atribut este **xmlns:prescurtare**=URI. Secvența **prescurtare** este folosită în cadrul documentului XML pentru a realiza o legătură între elemente și marcajele care le definesc. Așa cum am mai spus URI-ul este doar un șir unic folosit pentru a identifica zone pentru nume corespunzătoare. Chiar dacă acest șir este de fapt un URL, acesta este folosit doar ca nume unic și nu implică nici o conexiune la server-ul Web corespunzător. Acest URL reprezintă doar o modalitate de a identifica în mod unic o zonă pentru nume; puteți folosi în locul acestuia chiar și denumirea sortimentului de ciocolată preferat.

În cadrul elementelor XML propriu-zise vom prefixa marcajul cu șirul **prescurtare** (urmat de caracterul '**:**') corespunzător zonei pentru nume, căreia îi aparține elementul.

În exemplul anterior primul element <**ABONAT**> aparține zonei pentru nume `colab`, iar al doilea zonei `abonati`.

Zonele pentru nume pot fi folosite și pentru atributele elementelor XML așa cum se poate vedea în exemplul următor:

```
<COLABORATORI xmlns:colab=
              "http://www.ginfo.ro/colab"
  xmlns:abonati=
            "http://www.ginfo.ro/abonati">
  <COLABORATOR>
    <ID colab.tip="numeric">1001</ID>
    <colab:ABONAT>Yes</colab:ABONAT>
    <abonati:ABONAT>Oroian Tudor
    </abonati:ABONAT>
  </COLABORATOR>
</COLABORATORI>
```

În acest caz zona pentru nume este folosită pentru atributul **tip** al elementului <ID> pentru a putea diferenția acest atribut de alte atribute care au aceeași denumire, dar provin dintr-o altă zonă sau pentru care nu a fost specificată o astfel de zonă.

Modelul Document Obiect

Modelul Document Obiect (engl. *Document Object Model* - DOM) este folosit pentru a defini structura logică și modul de accesare al documentelor XML, HTML etc. DOM-ul are o importanță deosebită deoarece definește un mod standard de accesare și manipulare a documentelor XML.

Să considerăm următorul exemplu:

```
<AUTORI>
  <AUTOR>
    <ID>1001</ID>
    <NUME>Oltean</NUME>
    <PRENUME>Mihai</PRENUME>
  </AUTOR>
  <AUTOR>
    <ID>1002</ID>
    <NUME>Giumale</NUME>
    <PRENUME>Cristian</PRENUME>
  </AUTOR>
</AUTORI>
```

Documentele XML sunt ierarhice prin natura lor, adică există întotdeauna un nivel superior (elementul rădăcină) care are unul sau mai multe elemente fii, care la rândul lor pot avea fii și așa mai departe. Deci un document XML poate fi reprezentat printr-un arbore; arborele corespunzător documentului XML este prezentat în figura 1.

În termenii modelului document obiect elementele poartă și denumirea de *noduri*, un nod reprezentând un element în această structură arborescentă.

Obiecte de bază

Pentru reprezentarea acestei structuri ierarhice, DOM oferă o colecție de obiecte, metode și proprietăți care permit manipularea acestora.

Principalele obiecte sunt `Node` care reprezintă un nod din această ierarhie, `NodeList` care reprezintă o colecție de noduri și `NamedNodeMap` care reprezintă o colecție de noduri care pot fi accesate atât prin intermediul numelor, cât și al indicilor.

Există și câteva metode care permit parcurgerea nodurilor. Metoda `childNodes` returnează o listă (`NodeList`) care conține fiii unui nod. Prin intermediul metodei `firstChild` putem obține primul fiu al unui nod, ultimul fiu putând fi obținut cu ajutorul metodei `lastChild`. Metoda `parentNode` poate fi folosită pentru a afla părintele unui nod, în timp ce metodele `previousSibling` și `nextSibling` sunt folosite pentru a determina fratele din stânga (nodul anterior de pe același nivel al ierarhiei), respectiv fratele din dreapta al unui nod (următorul nod de pe același nivel al ierarhiei). În sfârșit, metodele `nodeName` și `nodeValue` sunt folosite pentru a

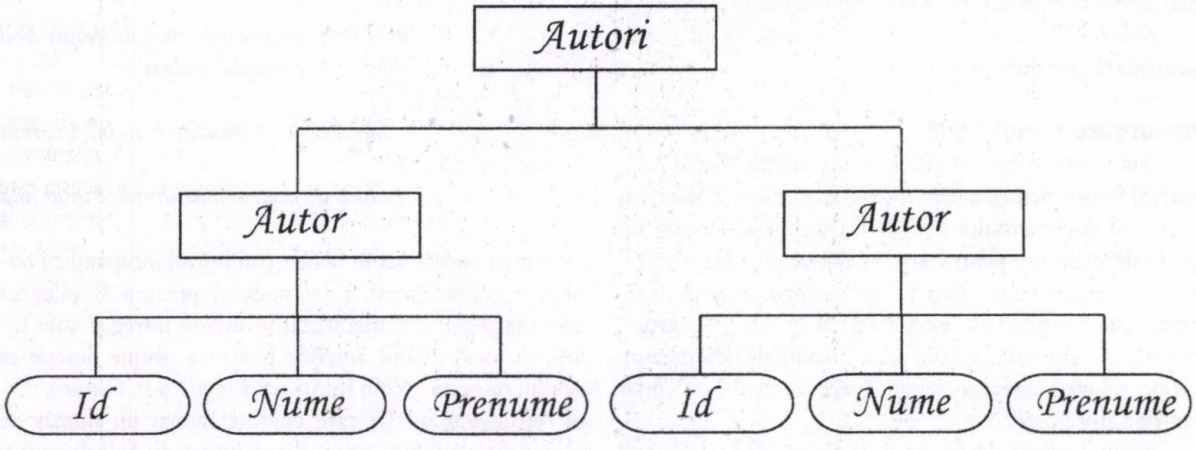


Figura 1: Modelul Document Obiect

determina numele, respectiv valoarea corespunzătoare unui nod.

Obiecte specifice ale unui DOM

Limbajul XML a fost astfel proiectat încât să fie ușor extensibil și să acopere o mare varietate de documente. Din acest motiv există obiecte specifice fiecărui tip de nod, majoritatea acestora moștenind proprietățile și metodele obiectului `Node` corespunzător. Vă prezentăm în continuare câteva dintre aceste obiecte.

Obiectul `Document` reprezintă obiectul rădăcină al unui document XML.

Obiectul `DocumentType` conține informații referitoare la DTD-ul sau schema asociată documentului XML și este echivalent cu construcția `!DOCTYPE` din DTD-uri.

Obiectul `DocumentFragment` reprezintă o copie a obiectului `Document` care poate fi folosit pentru inserarea de date sau pentru stocări temporare.

Obiectul `Element` reprezintă un element dintr-un document XML.

Obiectul `Attribute` (sau `Attr`) reprezintă un atribut al unui element dintr-un document XML.

Obiectul `Entity` reprezintă o entitate și este echivalent cu construcția `!ENTITY` din DTD-uri.

Obiectul `EntityReference` reprezintă o referință la o entitate.

Obiectul `Notation` reprezintă o notație și este echivalent cu construcția `!NOTATION` din DTD-uri.

Obiectul `CharacterData` este obiectul de bază pentru informațiile de tip text dintr-un document XML.

Obiectul `CDATASection` reprezintă o informație sub forma unui șir de caractere și este echivalent cu construcția `!CDATA` din DTD-uri.

Obiectul `Text` reprezintă textul conținut de un nod element sau un nod atribut.

Obiectul `Comment` reprezintă un element comentariu dintr-un document XML.

Obiectul `ProcessingInstruction` reprezintă o instrucțiune de procesare ca cele din secțiunile `<? ?>`.

Obiectul `Implementation` reprezintă detalii de implementare specifice aplicației.

Parcurgerea unui DOM

Vom construi în continuare un document HTML care poate fi folosit pentru a parcurge un document XML. Din interiorul documentului HTML va trebui să alocăm un spațiu de memorie pentru a stoca datele din documentul XML. Pentru aceasta vom folosi o construcție de tipul `<XML id="dsoData" src="AUTORI.XML"></XML>`. Prin această construcție vom crea o *insulă de date* care nu are nici un efect direct în pagina Web, ci este folosită doar pentru a stoca date.

Pentru a accesa datele din această insulă de date vom folosi un element `<SPAN>` într-o construcție de forma `<SPAN id="dom"></SPAN>`. Scopul nostru este de a folosi modelul document obiect pentru a extrage datele în format XML și să le afișăm. Va trebui să pornim de la nodul rădăcină, să găsim fiii acestui nod și să afișam detaliile referitoare la acest nod. Vom afișa numele nodului, tipul și valoarea acestuia. De asemenea, din moment ce nodul poate avea fii, vom repeta procedeul pentru acești fii, coborând un nivel în structura arborescentă, până când vom vizita toate nodurile arborelui. Deoarece aceasta este o operație simplă de parcurgere a unui arbore, vom folosi o funcție recursivă pentru a o implementa.

Una dintre informațiile pe care le vom afișa este tipul nodului. Tipul va fi codificat printr-un întreg dar, deoarece acesta nu oferă prea multe informații, îl vom converti într-un număr care va descrie tipul nodului. Vom declara un vector global care conține descrierea tipurilor nodurilor și vom folosi ca indice numărul corespunzător tipului. În *JavaScript* acest lucru poate fi realizat astfel:

```
var Tipuri=new Array('',
  'ELEMENT (1)',
  'ATTRIBUTE (2)',
  'TEXT (3)',
  'CDATA SECTION (4)',
  'ENTITY REFERENCE (5)',
  'ENTITY (6)',
  'PROCESSING INSTRUCTION (7)',
  'COMMENT (8)',
  'DOCUMET (9)',
  'DOCUMENT TYPE (10)',
  'DOCUMENT FRAGMENT (11)',
  'NOTATION (12)');
```

Funcția recursivă pe care o vom crea va avea ca parametru un nod XML și un întreg care va indica nivelul pe care se află nodul. Acest al doilea parametru este folosit numai pentru a realiza indentarea care să permită o mai bună lizibilitate la afișare. Această funcție începe astfel:

```
function GasesteFii(nod, nivel){
  var strNod='';
        // șirul care conține informațiile despre nod
  var contor=0; // numărul nodurilor
  var indNod=0;
          // numărul corespunzător nodului curent
  var nodAtr;
            // lista de noduri a atributelor unui nod
```

Putem începe acum să construim șirul începând cu numele nodului curent, tipul nodului precum și valoarea acestuia. Tipul este identificat printr-un întreg și vom folosi vectorul definit anterior pentru a obține descrierea tipului respectiv. Vom folosi și o funcție `getIndent` care returnează un șir care conține numai un număr de blancuri corespunzător nivelului curent, șir folosit pentru indentare. Funcția `GasesteFii` continuă cu:

```
strNod += getIndent(nivel)
       + '<B>'+nod.nodeName
```

```
    + '</B>   Tip: <B>'
    + Tipuri[nod.nodeType]
    + '</B>   Valoare: <B>'
    + nod.nodeValue +'</B><BR>';
```

Urmează să verificăm dacă nodul curent are atribute și, dacă acestea există, să adăugăm în șirul `StrNod` informațiile referitoare la acestea. Vom completa procedura `GasesteFii` cu următoarele instrucțiuni:

```
nodAtr = nod.attributes;
if (nodAtr!=null){
  contor=nodAtr.length;
  if (contor>0){
    for (i=0;i<contor;i++)
      strNod += getIndent(nivel+1)
             + '<B>'
             + nodAtr(i).nodeName
             + '</B>   Tip: <B>'
             + Tipuri[nodAtr(i).nodeType]
             + '</B>   Valoare: <B>'
             + nodAtr(i).nodeValue
             + '</B><BR>';
    }
  }
```

În sfârșit va trebui să verificăm dacă nodul are fii și pentru fiecare fiu să apelăm recursiv funcția `GasesteFii`. Deci, funcția este completată de instrucțiunile:

```
contor=nod.childNodes.length;
if (contor>0){
  for (indNod=0;indNod<contor;indNod++)
    strNod+=GasesteFii
      (nod.childNodes(indNod), nivel+1);
  return strNod;
}
```

Pentru a putea afișa aceste informații în cadrul documentului HTML am putea adăuga liniile:

```
domXML=dsoData
dom.innerHTML=GasesteFii(domXML,0);
```

Vă prezentăm în continuare întregul document HTML care trebuie creat pentru a parcurge un DOM.

```
<HTML>
  <HEAD>
    <TITLE>
      Modelul Document Obiect
    </TITLE>
    <SCRIPT language="JavaScript">
      var Tipuri=new Array('',
        'ELEMENT (1)',
        'ATTRIBUTE (2)',
        'TEXT (3)',
        'CDATA SECTION (4)',
        'ENTITY REFERENCE (5)',
        'ENTITY (6)',
        'PROCESSING INSTRUCTION (7)',
        'COMMENT (8)',
        'DOCUMENT (9)',
        'DOCUMENT TYPE (10)',
        'DOCUMENT FRAGMENT (11)',
        'NOTATION (12)');

      function getIndent(nivel){
        var blanc='';
        for (i=0;i<nivel;i++)
          blanc+='     ';
        return blanc;
      }

      function GasesteFii(nod, nivel){
        var strNod='';
        var contor=0;
        var indNod=0;
        var nodAtr;
        strNod  += getIndent(nivel)
                + '<B>'
                + nod.nodeName
                + '</B>   Tip: <B>'
                + Tipuri[nod.nodeType]
                + '</B>   '
                + 'Valoare: <B>'
                + nod.nodeValue
                +'</B><BR>';
        nodAtr = nod.attributes;
        if (nodAtr!=null){
          contor=nodAtr.length;
          if (contor>0)
            for (i=0;i<contor;i++)
              strNod  += getIndent
                              (nivel+1)
                      + '<B>'
                      + nodAtr(i).
                                nodeName
                      + '</B>   '
                      + 'Tip: <B>'
                      + Tipuri[nodAtr
                          (i).nodeType]
                      + '</B>  '
                      + 'Valoare: <B>'
                      + nodAtr(i).
                                nodeValue
                      + '</B><BR>';
        }
        contor=nod.childNodes.length;
        if (contor>0)
          for (indNod=0;indNod<contor;
                              indNod++)
```

```
                strNod += GasesteFii(
                    nod.childNodes(indNod),
                                    nivel+1);
            return strNod;
        }
    </SCRIPT>
</HEAD>
<BODY>
    <XML id="dsoData"
                    src="AUTORI.XML"></XML>
    <SPAN id="dom"></SPAN>
    <SCRIPT language="JavaScript">
      domXML=dsoData;
     dom.innerHTML=GasesteFii(domXML,0);
    </SCRIPT>
</BODY>
</HTML>
```

Pagina Web afișată de programul de navigare *Internet Explorer 5.0* este prezentată în figura 2.

Se poate observa foarte clar natura recursivă a modelului document obiect. Pe nivelul 0 avem nod de tip `#document` care este nodul rădăcină pentru orice document XML. De asemenea, se observă faptul că acesta nu este un element propriu-zis, tipul său fiind `Document`. Deci rădăcina este întotdeauna un document XML, iar pe nivelurile inferioare avem elemente XML.

Primul element se află pe nivelul 1 și este elementul rădăcină al documentului XML (**AUTORI**). Acesta conține câte un element pentru fiecare **AUTOR** și câte un element pentru fiecare caracteristică a autorului. De asemenea, se observă că pentru fiecare frunză (nod fără nici un fiu) avem un alt nod numit `#text`. Acesta conține textul propriu-zis (valoarea nodului). Am ajuns la un punct care creează deseori confuzii. Se pune întrebarea: *de ce fiecare element are valoarea* `null` *și subelementul său numit* `#text` *conține de fapt valoarea nodului?, de ce valoarea nodului nu este acel text?* Răspunsul este destul de simplu; motivul este faptul că nodurile pot conține atât text, cât și alte noduri. În aceste cazuri ar trebui oricum să alegem pentru valoarea nodului fie textul, fie o referință spre unul dintre fii. Această ambiguitate care ar fi putut apărea a determinat *Consorțiul World Wide Web (W3C)* să decidă că textul corespunzător unui nod va fi "păstrat" într-un nod fiu al cărui tip este `TEXT`.

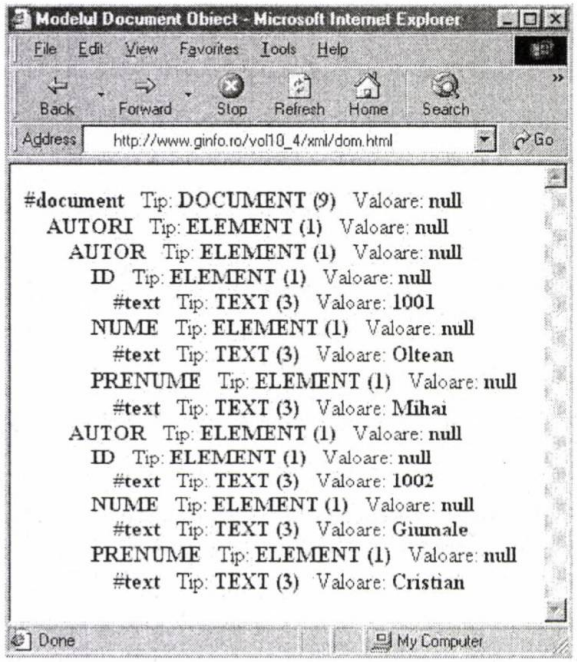


Figura 2: DOM.HTML în Internet Explorer 5.0

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro.

news...

Netscape 6.0

Netscape a lansat o versiune preliminară a programului de navigare Netscape, denumită Preview Release 1 of Netscape 6. Această nouă generație de browser Web bazată pe engine-ul de randare Gecko va dispune de o nouă interfață, codul va fi open-source și vor fi oferite diferite alte facilități. Deși numărul acestei versiuni ar fi trebuit să fie 5, compania preferat să sară peste acest număr, pentru a obține cel puțin un avantaj psihologic față de Microsoft.

Windows Me Beta 3

Microsoft a anunțat că Windows Me, succesorul sistemului de operare Windows '98 a ajuns la versiunea Beta 3. Aceasta nouă versiune are mai multe îmbunătățiri și include noile variante ale Microsoft Movie Maker, Media Player 7, update-uri de USB etc.

Această ultimă versiune Beta dinaintea lansării oficiale de pe 2 iunie 2000 se dovedește a fi mult mai stabilă decât versiunile anterioare.

Planuri pentru anul 2000

Mihai Scorțaru

Deși cursa spre bariera de 1 GHz a luat sfârșit cu victoria celor de la AMD, războiul procesoarelor continuă și, deocamdată, este cea mai spectaculoasă confruntare din domeniul IT. Vă prezentăm în continuare care sunt planurile pentru anul 2000 ale firmelor Intel și AMD.

Planurile Intel

Compania *Intel* plănuiește să lanseze versiunea la 933 MHz a procesorului *Pentium III* în cursul celui de-al treilea trimestru al acestui an. La ora actuală se pare că nu există planuri pentru versiuni la 900 și 950 MHz ale acestui procesor. Pentru a obține astfel de frecvențe cu o magistrală de 100 MHz multiplicatorul ar trebui să fie 9, respectiv 9,5 și se pare că *Intel* se confruntă cu probleme în obținerea acestor valori. Totuși, nou lansatul procesor *Intel Celeron* la 600 MHz are magistrala de 66 MHz, deci multiplicatorul este 9.

Succesorul procesoarelor *Pentium III Coppermine* este cunoscut în momentul de față doar sub numele de cod *Willamette*. Se presupune că acest procesor va fi lansat doar în ultimul trimestru al anului 2000 și, pentru început, va avea frecvențe de 1,3 și 1,4 GHz. Versiuni de 1,5 GHz și mai mult sunt așteptate numai la începutul anului 2001. Ca și în cazul saltului de la 800 la 1000 MHz, *Intel* este din nou criticată pentru că va sări de la 1 GHz direct la 1,3 GHz fără a "acoperi" golul rămas. Totuși s-ar putea ca aceasta să nu fie o problemă deoarece s-ar putea ca *Willamette* la 1,3 GHz să nu fie mai rapid decât *Coppermine* la 1 GHz.

Ultimele procesoare *Celeron* la 566 și 600 MHz folosesc, în sfârșit, miezul *Coppermine*, dar jumătate de *cache*-ul L2 este dezactivat. Această dezactivare, împreună cu păstrarea magistralei de 66 Mhz, asigură o diferență artificială de performanță între *Celeron* și *Pentium III*. În cursul celui de-al patrulea trimestru al acestui an *Intel* va lansa procesoare *Celeron* la 633, 667 și 700 MHz, iar versiuni la 733 MHz și mai mult sunt așteptate în primul trimestru al anului viitor.

Planurile AMD

Viitorul procesor *AMD* destinat calculatoarelor ieftine poartă numele de cod *Spitfire* și probabil va fi lansat la sfârșitul lunii mai sau începutul lunii iunie. Acest procesor reprezintă răspunsul direct al companiei *AMD* la procesorul *Celeron*, dar se pare că acesta va fi net superior procesorului de la *Intel*. S-ar putea ca *Spitfire* să depășească chiar și performanțele procesoarelor *Athlon* actuale. *Spitfire* este proiectat pe baza tehnologiei de 0,18 microni pe bază de aluminiu cu 64 KB *cache* L2 integrat care funcționează la viteza procesorului. Chiar dacă memoria *cache* L2 este de opt ori mai mică decât cea a procesoarelor *Athlon* actuale, performanțele ar putea fi superioare deoarece *cache*-ul L2 al procesoarelor *Athlon* este extern și funcționează doar la jumătate, două cincimi sau o treime din viteza procesorului.

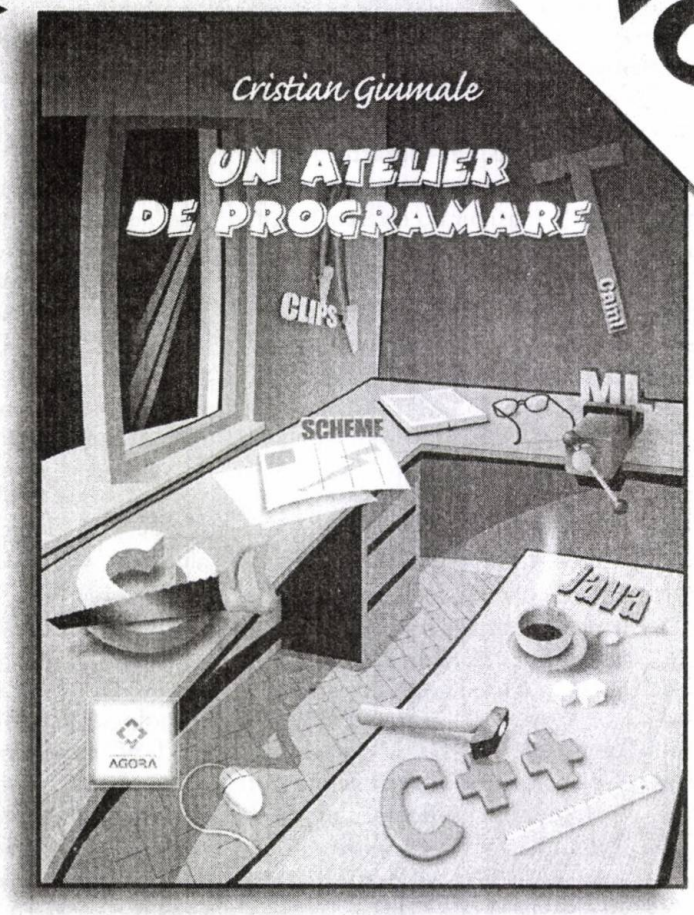
Succesorul procesorului *Athlon* are numele de cod *Thunderbird* și probabil va fi lansat în același timp cu *Spitfire* pentru a evita ca procesorul ieftin *Spitfire* să fie mai performant decât procesorul mai scump *Athlon*. Procesorul *Thunderbird* va avea pentru început viteze de minimum 1,2 GHz, dar s-ar putea ca *AMD* să producă și procesoare cu frecvențe mai mici care să înlocuiască procesoarele *Athlon* actuale. Procesoarele *Thunderbird* produse la Austin vor fi fabricate folosind tehnologia de 0,18 microni pe bază de aluminiu, vor avea 256 KB *cache* L2 integrat. Vor exista și procesoare *Thunderbird* produse la noua fabrică *AMD* de la Dresda; acestea vor fi fabricate tot folosind tehnologia de 0,18 microni, dar vor fi pe bază de cupru și s-ar putea să aibă un *cache* L2 de 512 KB.

Nu în ultimul rând, *AMD* plănuiește să lanseze un procesor destinat stațiilor *high-end* și serverelor, procesor al cărui nume de cod este *Mustang*. Acesta este bazat pe un miez *Athlon* îmbunătățit și are un *cache* L2 de 1-2 MB. Probabil că acest procesor nu va apărea pe piață decât în luna septembrie. Nu sunt disponibile prea multe informații despre acest procesor, dar se pare că va fi fabricat la Dresda și va profita de tehnologia de 0,18 microni pe bază de cupru.

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro.

COMPUTER LIBRIS AGORA

NOUTATE



Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

CONCEPȚIE, ARHITECTURĂ ȘI FUNCȚIONARE

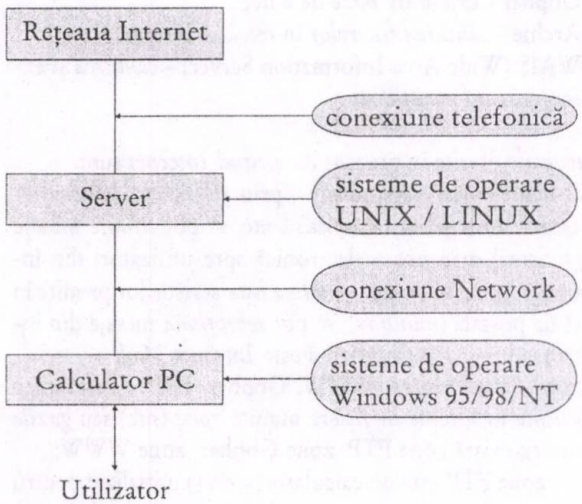
Marin Vlada

În cadrul episodului anterior am prezentat câteva detalii referitoare la apariția și funcționarea Internet-ului. Vom continua acum cu prezentarea modelului TCP/IP, precum și câteva aspecte referitoare la conectarea la Internet și utilizarea serviciilor oferite de această rețea.

Arhitectura și sarcinile TCP/IP

Structura arhitecturii TCP/IP este dată de *nivelurile funcțiilor* definite de *rețelele interconectate* în *mediul* Internet. *Modelul arhitectural* TCP/IP este conceput să controleze următoarele *niveluri de sarcini*:

- nivelul rețea - dat de un singur protocol care recunoaște toate *host-urile* și *nodurile* din *rețeaua* Internet;
- nivelul *Internet* - dat de *software* ce implementează conceptia *mediului* Internet, adică IP (*Internet Protocol*) și ICMP (*Internet Control Message Protocol*);
- nivelul comunicației - dat de *software* TCP (*Transmission Control Protocol*) și UDP (*User Datagram Protocol*);
- nivelul aplicației - dat de *sarcinile* invocate de *utilizatorii Internet*.



Nivelurile funcțiilor în arhitectura TCP/IP sunt următoarele:

- nivelul fizic - asigură transmiterea digitală a informațiilor prin mediul fizic de interconectare de tip telecomunicație (*rețea telefonică*, *rețea publică de date*, *sateliți de comunicații*);
- nivelul legatură de date - detectează și corectează *erorile de transmisie* produse la nivelul fizic;
- nivelul rețea - asigură *dirijarea pachetelor* într-o rețea cu legături de tip "*point to point*";
- nivelul transport - asigură *comunicarea* între două *calculatoare host* din rețea;
- nivelul sesiune - stabilește și întreține *conectări între procese* pentru transmiterea de date în rețea;
- nivelul prezentare - realizează *transformări ale reprezentării datelor*, comprimarea textelor, conversii de fișiere, prelucrarea și vizualizarea imaginilor, prelucrarea și redarea sunetelor;
- nivelul aplicației - realizează *comunicarea* pentru *schimbul de informații* între utilizatori.

Nivelul rețea

Protocoalele Internet la nivel de rețea *furnizează servicii* ce permit comunicarea dintre două noduri ale rețelei, asigurând *dirijarea* pachetelor de date ce provin de la protocoalele de la *nivelul de transport*.

La acest nivel are loc atât conversia pachetelor de date în UDP (*User Datagram Protocol*) și transmiterea lor către protocoalele nivelului legătură de date pentru procesarea lor ulterioară (la transmisie), cât și transferarea lor către nivelul transport, la recepție.

La *nivelul rețea* există următoarele *protocoale*:

- *protocolul* ARP (*Address Resolution Protocol*) - translatează dinamic *adresele Internet* în *adrese hardware* unice în cadrul rețelelor locale;

- *protocolul* RARP (*Reverse Address Resolution Protocol*) - tranformă invers *adresele hardware* corespunzătoare nodurilor în *adrese* **Internet**;
- *protocolul* ICMP (*Internet Control Message Protocol*) - permite *schimbul de informații de control* între nodurile rețelei (*host*-uri, *router*-e, *gateway*-uri);
- *protocolul* IP (*Internet Protocol*) - *asigură servicii de transmitere* a pachetelor de date recepționate de la protocoalele nivelului transport.

Nivelul transport

Nivelul transport este implementat în *rețelele* TCP/IP prin intermediul protocoalelor UDP și TCP. *Sistemele de operare* Unix și Linux oferă programelor aplicative o interfață comună pentru aceste protocoale. *Protocoalele* TCP/IP de nivel transport oferă servicii ce permit programelor nivelului aplicației să comunice între ele prin intermediul *mesajelor*.

Nivelul aplicației

Nivelul cel mai înalt este *nivelul aplicației* ce utilizează *serviciile* protocoalelor de la nivelul transport pentru a satisface cerințele utilizatorilor *mediului* Internet.

Familia de procoale TCP/IP este operațională deoarece aceste protocoale sunt *gestionate* prin *numere speciale de porturi* (*well-known numbers*) și *nume speciale de protocoale și servicii* (*well-known names and services*).

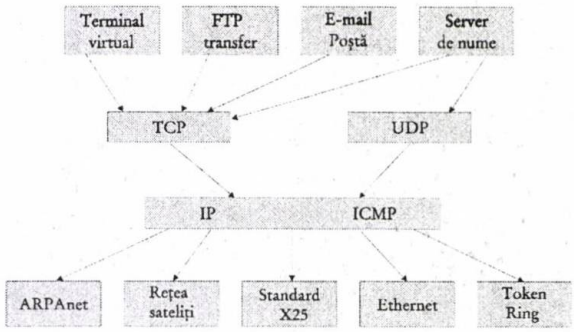
Principalele *instrumente* **Internet** la *nivel de aplicație* sunt:

- E-mail (*Electronic Mail*) - *poșta electronică* (trimiterea și primirea de mesaje);
- Telnet - *protocol pentru conectarea la distanță*;
- FTP (*File Transfer Protocol*) - *protocol pentru transfer de fișiere*;
- HTTP (*Hyper Text Transfer Protocol*) - *protocol pentru transfer de hypertext*;
- EGP (*Exterior Gateway Protocol*) - *protocol pentru gateway exterior*;
- Domain (*Domain Name Protocol*) - *protocolul numelui de domeniu*;
- Login (*Remote Login Protocol*) - *protocolul de login la distanță*;
- RIP (*Routing Information Protocol*) - *protocolul informațiilor de rutare*;
- ICMP (*Internet Control Message Protocol*) - *protocolul de control al mesajelor*.

Aspecte de conectare la Internet

Există două *modalități principale de conectare fizică* la un *server* din *sistemul* **Internet**:

- conexiune telefonică PPP (*Point-to-Point Protocol*) - prin intermediul unui *modem*; în acest caz firma furnizoare de *servicii* **Internet** va da utilizatorului un *cont*, o *parolă* și un *număr de telefon* prin care se realizează conectarea;
- conexiune Network - prin intermediul *conectării de tip rețea* (cablu coaxial) la un *server* conectat deja la *sistemul* **Internet**;



Arhitectura conectării la sistemul Internet

E-mail și comunicare în Internet

Comunitatea rețelelor de calculatoare interconectate în *sistemul* **Internet** își asigură realizarea serviciilor prin intermediul protocoalelor ce au ca suport *sistemele de operare* ***Unix***, ***Linux*** sau ***Windows*** sisteme ce oferă facilități moderne pentru *comunicarea în rețea* (*servicii de interconectare*):

- Telnet - *limbaj de comunicație în rețea; conectarea la distanță*;
- E-mail (Electronic Mail) - *poșta electronică (emiterea și recepția scrisorilor);*
- FTP (File Transfer Protocol) - *transfer de fișiere*;
- HTTP (HyperText Transfer Protocol) - *legătura între informații aflate la distanță.*

Mediul **Internet** utilizează si *instrumente* (sisteme hipertext) concepute special pentru *localizarea*, *căutarea* și *vizualizarea* în rețea, cum ar fi:

- WWW (World Wide Web - pânza mondială) - *sistem de căutare și vizualizare*;
- Gopher - *explorare baze de date*;
- Archie - *căutarea fișierelor* în *rețeaua* Internet;
- WAIS (Wide Area Information Server) - *căutarea și extragerea informațiilor.*

Serviciile oferite în prezent de *mediul* **Internet** sunt:

- E-mail (*poșta electronică*) - prin utilizarea *comenzilor* ***Unix/Linux*** `pine`, `elm`, `mail` etc. se pot *trimite mesaje* (scrisori) prin poșta electronică spre utilizatori din întreaga lume, se poate *vizualiza lista* scrisorilor primite în cutia poștală (*mailbox*), se pot *recepționa* mesaje din întreaga lume; o altă variantă este ***Internet Mail***;
- zone de informare (WWW, Gopher, FTP) - *informații stocate* în *sisteme de fișiere* numite *zone* (*site*) sau *gazde* (*host*); există zone FTP, zone Gopher, zone WWW;
  - zona FTP este un calculator(*server*) inițializat pentru conectarea în sistem, *consultarea și copierea de fișiere;*
  - zona Gopher conține *baze de date* de referințe încrucișate despre documente din întreaga lume;

- zona WWW conține *pagini Web* ce pot fi vizualizate cu *programe speciale de consultare* (*browser*), cum ar fi *Mosaic*, *Netscape*, *Internet Explorer*; *paginile Web* pot conține trimiteri la alte zone WWW la care este acces, execuția se face prin *click* cu butonul mouse-ului;

- rețeaua UseNet - folosind un *cititor de știri* (newsreader), cum ar fi `rn`, `trn`, `nn`, sau `tin`, se pot studia mesaje din *grupurile de știri* (newsgroups) pe orice temă: *știință, medicină, economie, sport, politică, muzică, calculatoare etc.*

Accesul la Internet prin Telnet sau ssh

Definiție

Programul (serviciul) Telnet *reprezintă modalitatea de conectare la distanță, și anume accesarea unui server la care utilizatorul are un cont sau a unui server la care utilizatorul se poate conecta printr-un login cunoscut cu/fără parolă și care oferă posibilitatea utilizatorului să consulte biblioteci, baze de date etc.*

Astăzi, există diverse *implementări* Telnet pentru sistemele de operare *Dos*, *Windows*, *Unix* sau *Linux*.

Definiție

Serviciul ssh *realizează accesarea unui server la care utilizatorul are un cont și o parolă și care oferă posibilitatea utilizatorului de a avea acces la toate serviciile oferite de mediul* Internet.

Comenzi și operații după accesarea serverului

După *accesarea serverului direct* (serverul la care utilizatorul are un cont și o parolă), utilizatorul poate folosi cunoștințele acumulate privind *sistemele de operare* ***Linux*** sau ***Unix***, pentru a afla *diverse informații* despre *organizarea serverului*, despre *directoarele/fișierele* aflate pe server, despre *utilizatorii recunoscuți* de server etc.

Exemple

- afișarea pe ecran a tuturor utilizatorilor conectați la momentul respectiv; se utilizează *comanda* `w` sau `who`
- schimbarea directorului curent în directorul rădăcină; se utilizează *comanda* `cd /`
- listarea numelor subdirectoarelor din directorul curent; se utilizează *comanda* `ls -l`

Utilizarea serviciului E-mail

Definiție

Serviciul E-mail *oferit de sistemul* Internet *realizează pentru orice utilizator conectat la sistem:*

- *gestiunea unei cutii poștale (*Mail Box*) de unde se pot trimite mesaje (scrisori) către orice utilizator Internet ce are propria lui cutie poștală (adresă E-mail) și unde se pot primi mesaje (scrisori) de la orice utilizator E-mail;*
- *operații asupra mesajelor din cutia poștală a utilizatorului pentru listarea mesajelor, vizualizarea conținutului mesajelor, ștergerea mesajelor etc.;*
- *protecția cutiei poștale și a mesajelor;*
- *atașarea la mesajele trimise a unor fișiere.*

Crearea unei cutii poștale

La început, *cutia poștală proprie* a unui utilizator este creată fie de utilizatorul respectiv, fie de *administratorul serverului.*

Observație

Un utilizator poate avea mai multe cutii poștale aflate pe diverse servere. De exemplu, autorul acestui articol are următoarele adrese:

- *pe serverul* `math`: vlada@math.math.unibuc.ro;
- *pe serverul* `gw-chimie`: vlada@gw-chimie.math.unibuc.ro;
- *pe serverul* `hotmail.com`: vladamarin@hotmail.com (post restant)

În cazul unei cutii poștale (*post restant*) accesibile prin WWW (*World Wide Web*), servicii oferite gratuit, de exemplu, de *serverele* :

- `http://www.hotmail.com`
- `http://www.iname.com`
- `http://www.forevermail.com`

utilizatorul își ceează singur cutia poștală reținând pentru utilizare *numele contului* și *parola*, în acest caz este asigurată *protecția la citirea corespondenței.*

Pentru crearea cutiei poștale (post restant) pe un astfel de server, se lansează în execuție un *program de navigare* (*Internet Explorer*, *Netscape Communicator* etc.), și se indică sau se selectează *adresa* (*locația* URL - *Uniform Resource Locators*) corespunzătoare fișierelor (paginilor Web) de pe server, de exemplu: **http://www.hotmail.com.** Pe site-ul corespunzător se află toate indicațiile necesare pentru a crea și utiliza căsuța poștală.

Citirea și trimiterea mesajelor

Software-ul pentru E-mail a fost conceput astfel încât *operația de citire a mesajelor* primite de un utilizator să se realizeze în *sistem de protecție* prin intermediul unui nume de utilizator (*login name*) și o parolă de conectare (*password*). Dacă la *nivelul serverului*, păstrarea corespondenței este *protejată*, în schimb la *nivelul operației de citire* protecția este realizată relativ *parțial*, ținând seama de *programul* cu care se realizează citirea mesajelor.

În cazul unei cutii poștale de pe un *server al unei rețele locale*, accesul la corespondență se poate realiza:

- sub *Linux/Unix* - folosind *comenzile* `pine` sau `mail`, după ce utilizatorul s-a *conectat* la server; *în acest caz protecția este asigurată*;
- sub *Windows* - folosind *programele specializate* (*Microsoft Outlook*, *Netscape Messanger* etc.); la prima utilizare trebuie să se realizeze o anumită *configurare de lucru*; ultimele versiuni ale acestor programe *încearcă* să rezol-

ve *protecția la citire a corespondenței*, în cazul în care la aceeași stație de lucru lucrează mai mulți utilizatori).

Comentariu

Deși sub Windows *serviciul* E-mail *ridică unele probleme de securitate (atunci când la aceeași stație lucrează mai mulți utilizatori), oferă în schimb diverse programe cu interfețe grafice eficiente și comode pentru citirea mesajelor, trimiterea de mesaje și fișiere, codificând și decodificând automat fișiere.*

Pentru consultarea unei cutii poștale (post restant), se lansează în execuție un *program de navigare* (*Internet Explorer, Netscape Navigator* etc.), și se indică sau se selectează *adresa* corespunzătoare paginii Web care oferă acest serviciu. Se va introduce numele de utilizator și parola, iar apoi aceste date vor fi trimise spre server) de obicei prin apăsarea unui buton). Va fi încărcată o nouă pagină Web prin intermediul căreia pot fi citite și trimise mesaje.

Domnul Marin Vlada este lector la Universitatea București și poate fi contactat la adresa vlada@math.math.unibuc.ro.

AMD K6-III+ 500

AMD a anunțat lansarea procesoarelor AMD K6 III+ destinate calculatoarelor portabile care funcționează la o frecvență de până la 500 MHz.

Aceste procesoare sunt realizate pe baza tehnologiei de 0.18 microni și oferă o serie de funcționalități remarcabile cum ar fi cache L2 de mare viteză, tehnologia de economisire a energiei PowerNow! și implementarea setului de instrucțiuni 3DNow! și a celui pentru prelucrarea semnalelor DSP.

Intel Celeron 600

Intel a lansat pe data de 29 Martie noile procesoare Celeron realizate pe baza tehnologiei de 0,18 microni. Această tehnologie permite atingerea unor viteze de procesare mai ridicate, un volum de producție mai mare și costuri generale de producție mai mici.

Noile procesoare Celeron sunt disponibile în variante de 600 MHz si 566 MHz, și sunt cele mai rapide procesoare Intel destinate calculatoarelor ieftine.

Profit la AMD

În cadrul unei conferințe de presă organizate în Sunnyvale, Carolina, AMD a făcut publice rezultatele financiare ale primului trimestru al anului 2000.

Cu această ocazie a fost anunțat un venit net record de 189.349.000$. Creșteri semnificative au fost înregistrate pe fiecare grup de producție: Grupul Produselor de Calcul, Grupul de Memorie și Grupul de Comunicații.

Totalul încasărilor a crescut cu 13 procente față de trimestrul anterior și cu 73% față de aceeași perioadă a anului trecut.

Profitul se datorează în primul rând succesului familiei de procesoare Athlon, procesoare care se dovedesc a fi superioare celor produse de compania rivală Intel. Reamintim că AMD a fost primul producător care a lansat pe piață procesoare pentru PC-uri cu frecvența de 1 GHz care funcționează la temperatura camerei.

Premii CERF 2000

Continuând tradiția inaugurată la edițiile anterioare CERF, echipele redacționale reunite ale publicațiilor editate de Computer PRESS Agora, decernează pentru a patra oară consecutiv "Premiile de tehnologie PC Report" la CERF 2000.

Distincțiile au fost acordate astfel: categoria Sisteme: Siemens GP7000M 600 de la Fujitsu Siemens Computers; categoria Echipamente mobile: NOKIA 7110; categoria Periferice: NSM 2000 JukeBox de la Star Print Trade Co SRL; categoria Imprimante: Xerox DocuPrint N2125; categoria Componente: placa de bază Asus K7V; categoria E-Echipamente de rețea: Apple AirPort; categoria Soft de bază: Microsoft Windows 2000; categoria Aplicații: Rapid Autonet de la Soft-Win; categoria Premieră: Compaq iPak.

Marele premiu CERF 2000 a fost acordat produsului Orinoco Wireless de la Lucent Technologies.

COMPUTER PRESS AGORA

PC REPORT

ECHIPAMENTE · SISTEME · APLICATII · RETELE · INTERNET

Vol. 9, Nr. 1. Ianuarie 2000 Lei 20.000

Din acest număr:

- **Unelte de modelare UML**
- **Introducere în Python**
- Ghid anti-hacker
- Solid Edge 7
- Standarde GNU
- XML prin SAX
- PC Concrete

www.pcreport.ro

Revista
specialiștilor
în tehnologia informației și comunicațiilor

nologia informației și comunicați

Cure-all?

Dacă vrei să dormi bine noaptea, încearcă Compaq Deskpro. El te eliberează de durerile de cap din timpul zilei, generate de managementul sistemelor de calcul, pentru că este furnizat cu utilitare şi facilităţi deosebite. Dintre acestea, Compaq Insight Manager LC este primul program creat de industria de profil pentru coordonarea unor grupuri de PC-uri printr-o simplă interfaţă bazată pe browser. Cu noile facilităţi te bucuri de o formă fizică perfectă şi mai mult chef de muncă pentru că nu mai pierzi inutil timp pentru depanări şi câştigi ore preţioase de lucru efectiv. Este stimulentul pe care ţi-l oferă valoarea produselor Compaq, este motivul pentru care Deskpro este cel dintâi. Pentru alte idei fortifiante, vizitează site-ul www.compaq.com/deskpro sau contactează-ne la telefon: +40-1-3033100.

intel inside
pentium® III

COMPAQ

prin distribuitorul autorizat

OMNILOGIC BGS
135, Bucureşti-Ploieşti Road,
Bucharest 1, Romania
PO BOX 18-131
Tel: +40-1-303 31 00
Fax: +40-1-303 31 64
www.omnilogic.ro

Intel, Intel Inside şi Pentium sunt mărci înregistrate ale Intel Corporation

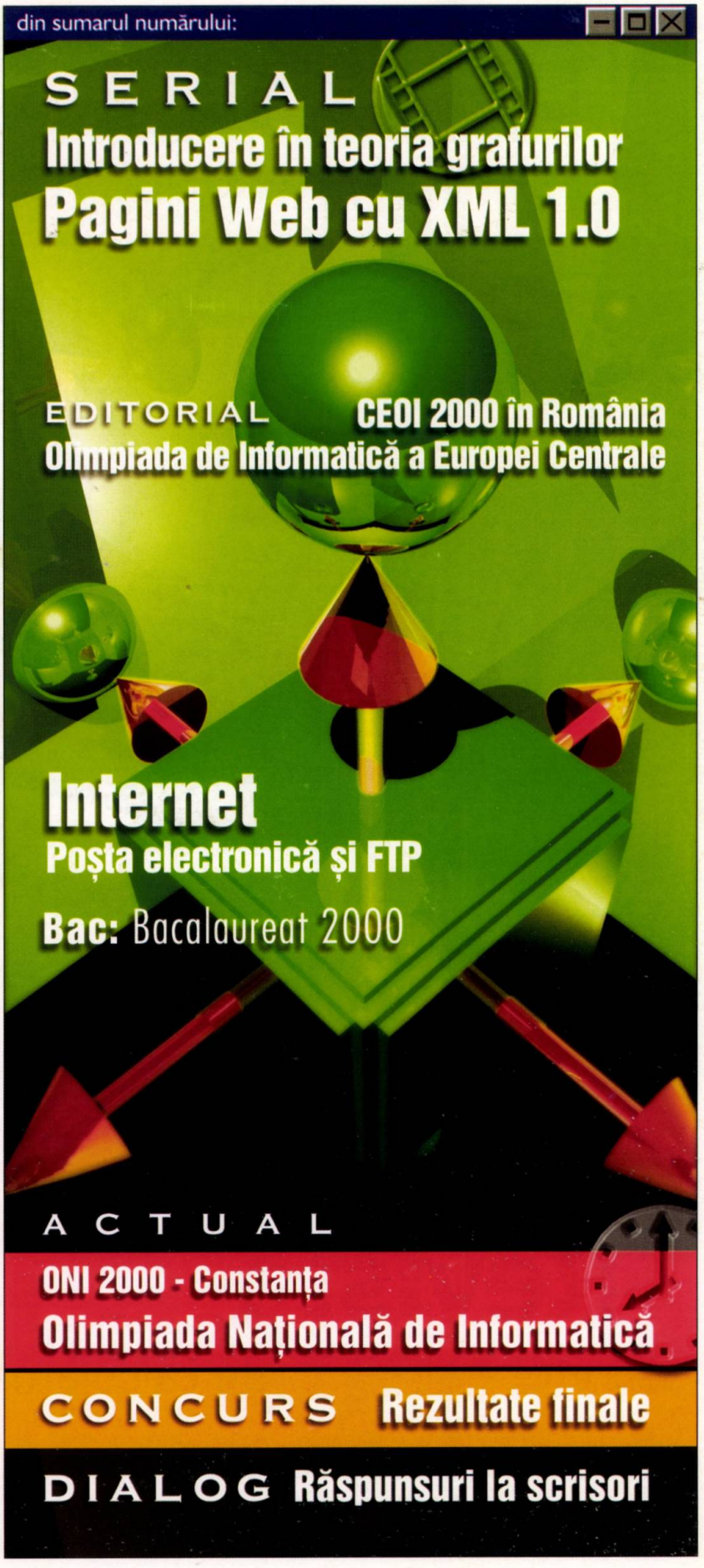
Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

Mai 2000
Vol. 10/5

Lei 15.000

din sumarul numărului:



SERIAL
Introducere în teoria grafurilor
Pagini Web cu XML 1.0

EDITORIAL **CEOI 2000 în România**
Olimpiada de Informatică a Europei Centrale

Internet
Poșta electronică și FTP

Bac: Bacalaureat 2000

ACTUAL
ONI 2000 - Constanța
Olimpiada Națională de Informatică

CONCURS **Rezultate finale**

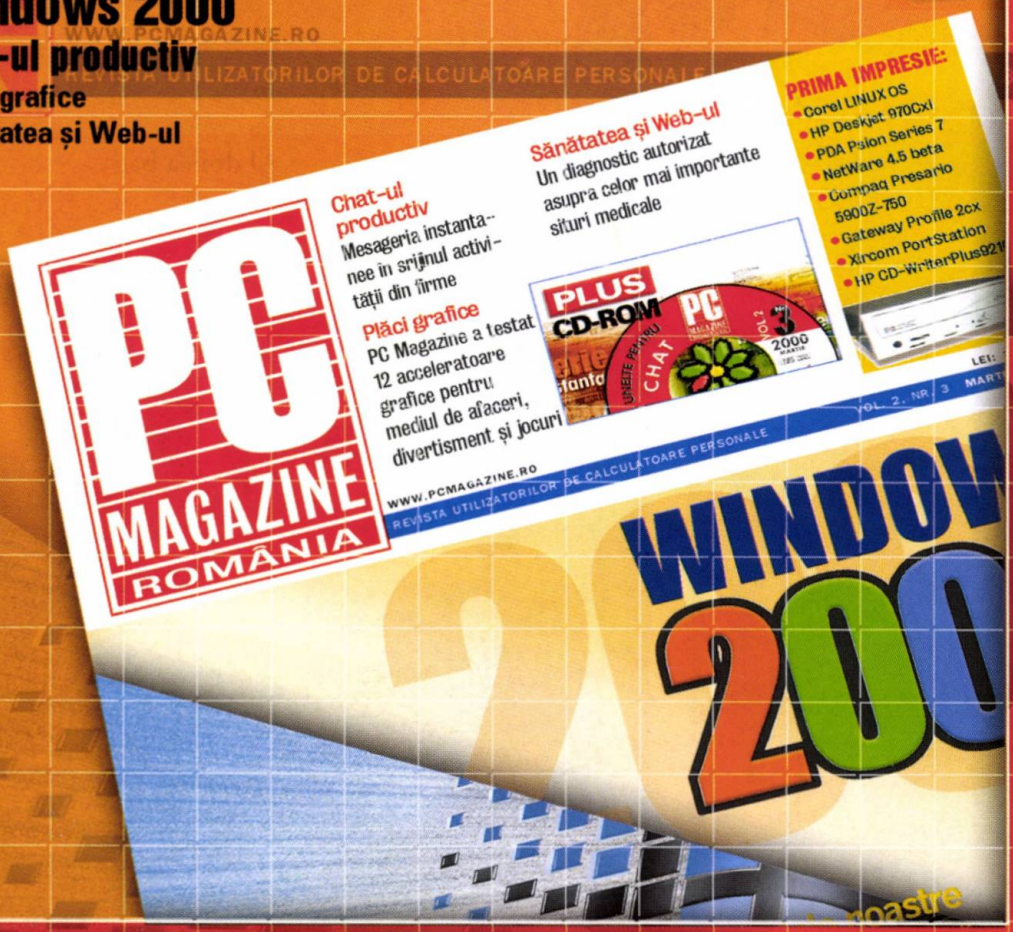
DIALOG **Răspunsuri la scrisori**

PC MAGAZINE ROMÂNIA

COMPUTER PRESS AGORA

Din acest număr:

- **Windows 2000**
- **Chat-ul productiv**
- **Plăci grafice**
- **Sănătatea și Web-ul**



www.pcmagazine.ro

Revista utilizatorilor de calculatoare personale

COMPUTER PRESS AGORA

Str. Tudor Vladimirescu nr. 63/1, Tg.Mureș (CP 230/1), 4300
Tel. 065-166516, Fax 065-166290, e-mail: office@agora.ro

Vol. 10, Nr. 5 /2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Cristian Cadar (București)
Marin Vlada (București)

Tehnoredactare:
Dan Crețu, Călin Borz
Corectura:
Carmen Bucur
Coperta și grafica:
Andrei Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Gabriela Bucșa (gbucsa@agora.ro)
Gabriel Piscupescu (gpiscup@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Anamaria Pascu (apascu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Mihaela Szabo (mszabo@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Sumar

editorial.. PAG. 3

CEOI 2000 în România

Olimpiada de Informatică a Europei Centrale

bac.. PAG. 30

BACALAUREAT 2000

Al doilea bacalaureat în informatică

actual.. PAG. 5

ONI 2000 - Constanța

Olimpiada Națională de Informatică

serial.. PAG. 36

Introducere în teoria GRAFURILOR
Cristian CADAR

Pagini WEB cu XML 1.0
Mihai SCORȚARU

probleme.. PAG. 7

propuse pentru REZOLVARE

Cele 24 de probleme de la ONI 2000

internet.. PAG. 45

POȘTA electronică și FTP

Marin VLADA

concurs.. PAG. 23

Rezultate finale

Rezultatele rundei a șasea

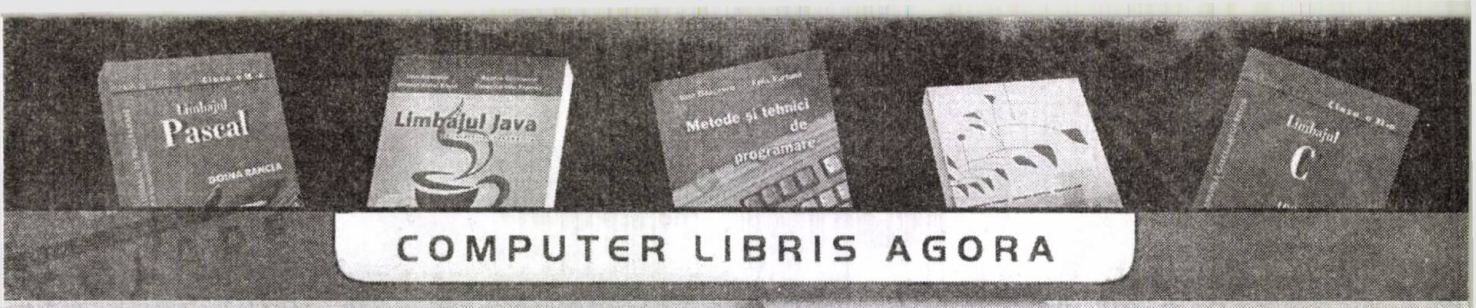
Rezolvări pentru runda a șasea

dialog.. PAG. 48

POȘTA redacției

Răspunsuri la scrisori

Așteptăm scrisorile voastre!



COMPUTER LIBRIS AGORA

NOUTATE

Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

CEOI 2000 în România!

Olimpiada de Informatică a Europei Centrale (*Central European Olympiad in Informatics*) se organizează anul acesta în România. Și nu întâmplător; alegerea (poate chiar - presiunea) Comitetului Olimpic al CEOI de a avea ediția din acest an "rotund" în țara noastră are mai multe explicații.

Stau în fața (nu, nu a hârtiei albe, ci a "monitorului alb") și caut cuvintele potrivite pentru a exprima simplu, ceea ce am simțit, de multe ori, împreună cu participanții din țara noastră, la alte olimpiade.

Cred că totul a început la Minsk, cu ocazia primei Olimpiade Internaționale de Informatică la care a participat și delegația României, condusă de domnul *prof. dr. Horia Georgescu*. Concurenții de atunci s-au făcut observați, delegația țării noastre a avut o contribuție științifică remarcată. Anii au trecut și azi, România este considerată țara care trimite concurenți foarte bine pregătiți la aceste olimpiade. S-a întâmplat nu de mult (în Turcia, la IOI, în timpul de lucru al concurenților), ca exprimându-mi îngrijorarea pentru "băieții" din echipă, conducătorul delegației Germaniei, încerca să mă liniștească: "...concurenții din România sunt programatori profesioniști, nu sunt motive de neliniște. Veți fi în top, ca de obicei..."

Deci, primul motiv ar fi *încrederea în valoarea științifică a învățământului românesc de informatică*, care astăzi are o largă recunoaștere internațională. Veți spune că fără talentul și pasiunea copiilor noștri nu s-ar face nimic... Ați avea dreptate doar parțial. Ca în orice grădină, și cea mai frumoasă floare se usucă, dacă nu este udată..., pe de altă parte cu o îngrijire zilnică se pot face minuni.

Revenind la olimpiade, această încredere a crescut de-a lungul anilor. În Argentina în 1993, *Radu Lupșa* a câștigat "aurul absolut".

În anul următor s-a organizat primul CEOI, în România, la *Liceul de Informatică* din Cluj. Niciodată nu vom uita "lupta" grea și munca pe care am depus-o (în echipele de organizare și comisiile științifice) pentru ca musafirii noștri să plece cu o imagine a României frumoasă și cu amintiri plăcute. Dar adevărata bucurie o simțim azi, constatând că acest concurs a fost dus mai departe de țările membre CEOI, organizându-se mereu, începând cu 1994, în fiecare an, în altă

țară. Organizatorii își amintesc de cei care au inițiat-o și reprezentanții României sunt "sărbătoriți" din an în an. În Polonia, domnul prof. *Andrzej Walat* a numit primul CEOI nu doar o surpriză plăcută, ci acel eveniment care greu se va întrece în ceea ce privește "farmecul" lui. Dar, evident nu farmecul este calitatea pentru care echipa Statelor Unite vine, ca țară invitată începând din 1997 la fiecare CEOI. Standardele științifice, nivelul concursului este considerat cel mai "tare" din toate concursurile de informatică din lume.

Așteptările ne obligă. Mai ales, din cauza rezultatelor României în anii care au urmat. Nu le voi înșira, am mai scris despre acestea în *GInfo*. Dar nu se poate uita că în 1999 prin *Mihai Stroe*, România din nou a câștigat un aur absolut la Olimpiada Internațională de Informatică, și că anul trecut *Mihai Pătrașcu* și *Ștefan Radu* au câștigat două medalii de aur la CEOI.

La Olimpiada Internațională de Informatică în toți acești ani echipa noastră s-a situat printre primele echipe din lume. În 1998 am fost întrebați dacă nu suntem doritori să organizăm IOI în 2007? În 1999 România a fost propusă în Comisia Internațională de Consiliere a IOI...

Dar, deocamdată: CEOI 2000 se desfășoară în România, în condițiile în care ultima olimpiadă internațională a fost organizată ireproșabil în Turcia, la standarde științifice și organizatorice neatinse până în 1999.

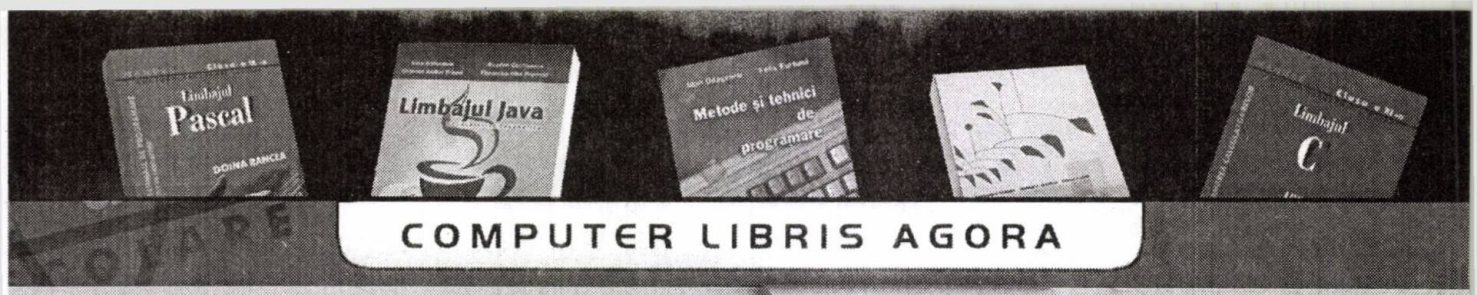
Și invitații noștri vin în România, după ce în Turcia au văzut că se pot oferi rezultate listate tuturor concurenților după două ore de la expirarea timpului de lucru. Au văzut că tot concursul s-a desfășurat conform planificării, respectându-se ora anunțată la toate activitățile.

Dar ce este și mai frumos, ei, organizatorii nu erau niciodată stresați, nervoși, nu i-a văzut nimeni "alergând". Așa ceva este posibil, doar dacă totul se pune la punct din timp. Deci, trebuie să tragem concluziile și să nu uităm nici un detaliu.

Scenariul CEOI 2000 există din decembrie 1999, dar mai sunt o serie de amănunte la care trebuie să lucrăm.

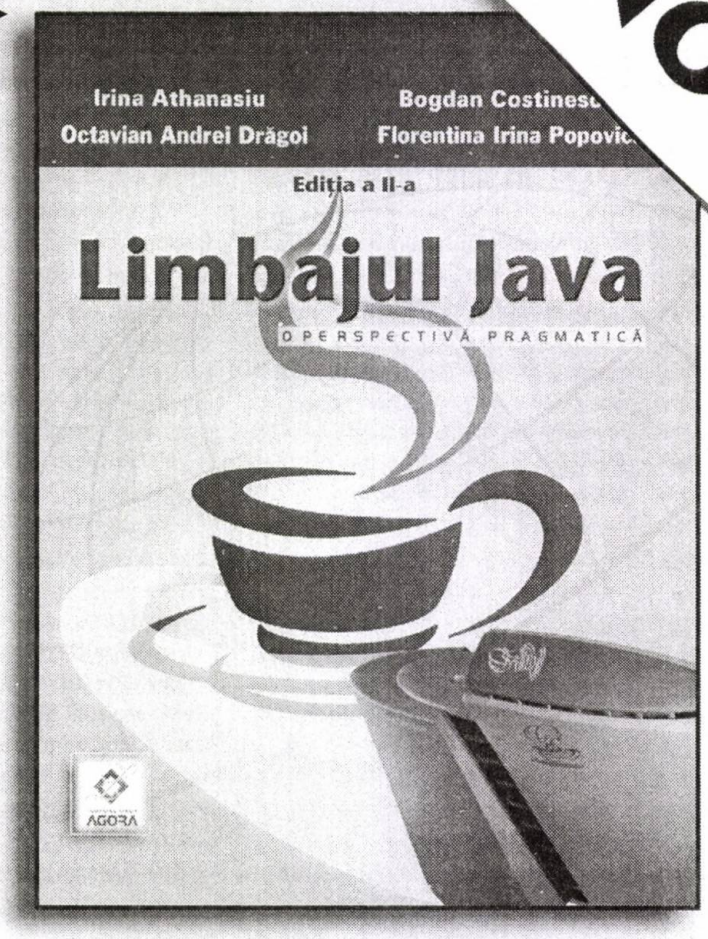
Urați-ne noroc și spor! Ne "revedem" la toamnă când vă voi "povesti" ce am reușit să realizăm din ceea ce am planificat...

Clara Ionescu



COMPUTER LIBRIS AGORA

NOUTATE



De ce Java ?
Deoarece tot ce se întâmplă acum în lumea calculatoarelor este pentru Internet, legat de Internet sau accesibil din Internet. Java este limbajul de programare care a devenit un standard de-facto în toate aceste cazuri. Java este acum mai mult decât un limbaj de programare, este o colecție de tehnologii pentru Internet.
De ce reeditată ?
Datorită dinamicii evoluției limbajului, autorii au considerat necesară verificarea exemplelor și pe versiunea Java 2 precum și adăugarea a două noi capitole: componente Java Beans și
securitate.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

ONI 2000 – Constanța

Clara Ionescu

Între 3 și 10 mai 2000 s-a desfășurat, la Constanța, cea de-a XXV-a ediție a Olimpiadei Naționale de Informatică. Ca de obicei, sarcina grea a organizării a fost îndeplinită de Ministerul Educației Naționale prin *Inspectoratul Școlar Județean Constanța* și liceele care au oferit spațiu, calculatoare și mai ales echipe de organizatori, formați din inspectori, profesori de liceu și elevi. Mulțumim tuturor celor care au stat alături de noi (participanți, concurenți și Comisia Centrală) și ne-au oferit cu ospitalitate condiții de concurs în școlile lor, (*Colegiul Național "Mircea cel Bătrân"*, *Liceul Teoretic "Ovidius"* și *Colegiul Pedagogic "Constantin Brătescu"*), dar și o ședere minunată pe litoral, ținând cont de faptul că elevii și profesorii însoțitori au fost cazați la Mamaia la hotelul Flora. Sponsori din județ, din București și din alte localități ale țării au oferit premii substanțiale câștigătorilor. Le mulțumim, și sperăm că nu ne vor uita nici în viitor.

Ca la orice Olimpiadă Națională, s-au adunat elevi și profesori din întreaga țară. 230 de concurenți au venit la Constanța în speranța câștigării unui premiu și în speranța de a fi selectați în lotul național de informatică. Dar nu pot obține premii toți participanții. Atunci nu ar mai fi concurs, nu-i așa? Partea științifică a fost susținută de cei 24 membri ai Comisiei Centrale care a propus în total 24 de probleme și a asigurat software-ul necesar evaluării automate a lucrărilor concurenților. A fost o muncă foarte grea, mai ales datorită faptului că zilele de concurs au fost planificate una după alta, iar premierea în a treia zi. Nu au lipsit nici incidentele. Într-una din zile ne-am "luptat" cu viruși, în cealaltă am avut surpriza de a fi "hackeriți"... Și nu am vorbit de emoția cu care am testat un sistem de evaluare nou, pregătit pentru CEOI 2000, care aici, la Constanța a fost "aruncat" prima dată "în apă adâncă". Bineînțeles, s-a dovedit a fi o testare extrem de utilă. Datorită faptului că, mai ales elevii de clasa a IX-a și a X-a, au reușit să provoace blocarea acestui sistem, (evident, fără să-și propună acest lucru, ci datorită unor erori de programare neobișnuite, sau mai bine zis neprevăzute de realizatorii sistemului) urmează să studiem motivele, și sistemul va fi "blindat".

La această olimpiadă concurenții au primit și un regulament din care ar fi trebuit să "învețe" câteva lucruri simple, dar din păcate nu toți concurenții au luat în serios sfaturile primite. Anul viitor sperăm să mai "crească" un pic, de asemenea, ar fi bine dacă acest regulament li s-ar difuza împreună cu ecusoanele sau alte materiale pe care le oferă gazdele la sosire.

Toți concurenții care au obținut premii (și mențiuni la clasele XI-XII) în urma celor două zile de concurs au avut dreptul să participe la cele două probe de baraj care s-a organizat în scopul stabilirii lotului național lărgit. La aceste probe, problemele au fost aceleași pentru toți concurenții, indiferent de clasă. Lotul s-a format din 15 elevi, după cum urmează.

Anul acesta primul concurs internațional, Balcaniada de Informatică s-a organizat în Macedonia între 15-20 mai. Primii patru concurenți din grupa claselor XI-XII au fost delegați pentru Balcaniadă și fac parte din lotul de informatică, fără să fi participat la baraje.

Mihai Pătrașcu (XI) - Dolj
Florin Ghețu (XI) - Vrancea
Angel Proorocu (XII)- București
Tudor Ioan Leu (XII) - Vrancea

În urma desfășurării celor două probe de baraj au fost selecționați ceilalți 11 membri ai lotului:

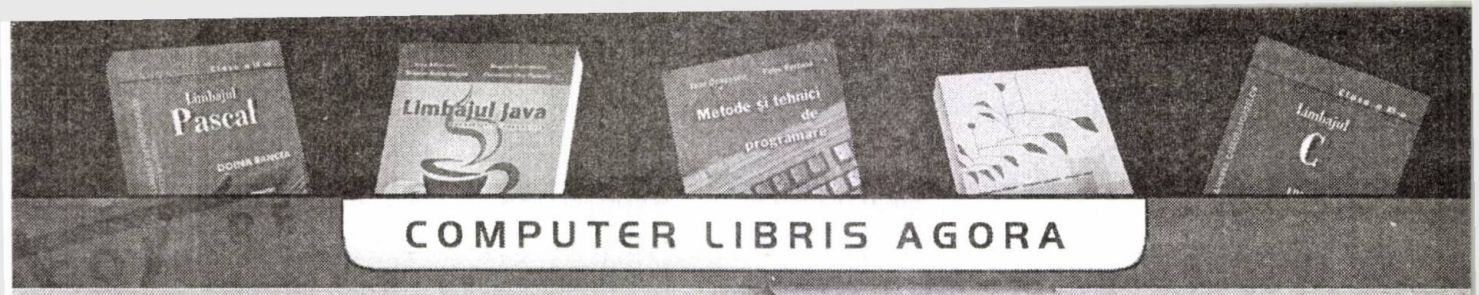
Ioana Ileană (XII) - Alba
Radu Ștefan (XII) - Brașov
Marius Andrei (XII) - București
Daniel Dumitran (X) - București
Mugurel Andreica (XI) - București
Vlad Dascălu (X) - Bacău
Liviu Păunescu (X) - Constanța
Alexandru Popa (XII) - București
Vencel Bors (X) - Bihor
Daniel Oprean (XII) - Alba

Îi felicităm și le dorim succes în continuare!

De asemenea, felicitări tuturor premianților, tuturor participanților și profesorilor lor, le dorim spor în dobândirea secretelor algoritmilor și a programării! Lista premianților o găsiți la pagina 47.

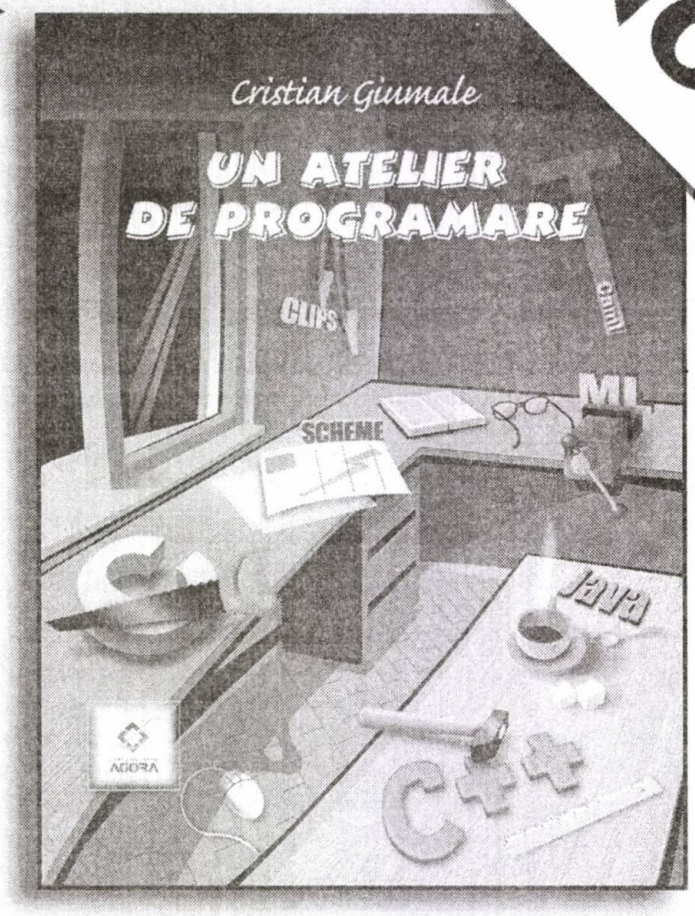
Nu putem să nu remarcăm că mai mult de jumătate din componenții lotului lărgit participă la concursul *Bursele Agora* organizat de ***GInfo.***

Clara Ionescu este redactor șef GInfo. Poate fi contactată la adresa: clara@cs.ubbcluj.ro



COMPUTER LIBRIS AGORA

NOUTATE



Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

AGORA

PROBLEME propuse pentru REZOLVARE

Olimpiada Națională de Informatică s-a organizat anul acesta la Constanța. Concursul s-a desfășurat pe trei grupe: clasa a IX-a, clasa a X-a și clasele a XI-a și a XII-a. De asemenea a existat o probă de baraj pentru selecționarea membrilor Lotului Național de Informatică. Vă propunem să încercați să rezolvați problemele cu care s-au confruntat concurenții la această competiție.

Clasa a IX-a

P050001: Algor

prof. Roxana Tâmplaru, Liceul "Ștefan Odobleja", Craiova

Georgel scrie un algoritm care conține structuri de atribuire, alternative, de selecție, repetitive și compuse. După scrierea algoritmului vrea să-l testeze pentru toate cazurile posibile. Pentru aceasta trebuie să cunoască numărul minim de date de test necesare.

Pentru descrierea structurilor se utilizează următoarele cuvinte cheie:

- pentru atribuire: `ATRIB`
- pentru alternativă: `DACA`
  `ATUNCI`
  `ALTFEL`
  unde `ALTFEL` poate lipsi
- pentru selecție: `ALEGE`
  `CAZ`
  `CAZ_IMPLICIT`
  unde `CAZ_IMPLICIT` poate lipsi
- pentru repetitive: 1) `CAT_TIMP`
  2) `REPETA`
  `PANA_CAND`
  3) `PENTRU`
- pentru compusă: `INCEPUT`
  `SFARSIT`

Nu se face diferență între literele mari și literele mici. Un algoritm începe cu cuvântul cheie `INCEPUT`, se termină cu `SFARSIT` și nu conține structuri vide. De asemenea, o structură compusă începe cu cuvântul cheie `INCEPUT` și se termină cu `SFARSIT`.

Cerință

Să se calculeze numărul minim de date de test necesare pentru verificarea algoritmului.

Date de intrare

Fișierul **`ALGOR.IN`** conține algoritmul, pe fiecare linie existând un cuvânt cheie. Nu există linii vide, algoritmul respectă principiile programării structurate și este corect.

Date de ieșire

În fișierul **`ALGOR.OUT`** se va scrie pe prima linie numărul minim de date de test necesare pentru verificarea algoritmului. Verificarea se bazează pe principiul "cutiei transparente", ceea ce înseamnă că testele se compun astfel încât să fie posibilă executarea algoritmului pe toate ramurile posibile. De exemplu, în cazul unei structuri repetitive `CAT_TIMP` care conține în corpul său un singur `ATRIB`, un test vizează o execuție fără să se intre în corpul structurii, altul pentru a trece cel puțin o dată și prin corpul acesteia. În mod similar se tratează și structura `PENTRU`.

Restricție

După cuvintele cheie `ATUNCI`, `ALTFEL`, `CAZ`, `CAZ_IMPLICIT`, `CAT_TIMP`, `REPETA`, `PENTRU` trebuie să existe obligatoriu o structură de atribuire, alternativă, de decizie, repetitivă sau compusă.

Exemplul 1

ALGOR.IN	ALGOR.OUT
INCEPUT	2
atrib	
DACA	
atunci	
ATRIB	
SFARSIT	

Exemplul 2

ALGOR.IN	ALGOR.OUT
INCEPUT	1
ATRIB	
REPETA	
inceput	
atrib	
atrib	
SFARSIT	
SFARSIT	

Observație

REPETA se execută cel puțin o dată.

Exemplul 3

ALGOR.IN	ALGOR.OUT
INCEPUT	3
ATRIB	
ALEGE	
CAZ	
ATRIB	
CAZ	
INCEPUT	
ATRIB	
ATRIB	
SFARSIT	
SFARSIT	

Observații

- se execută ATRIB de la primul CAZ;
- se execută ATRIB de la al doilea CAZ;
- nu se execută nici primul CAZ, nici al doilea CAZ.

Exemplul 4

ALGOR.IN	ALGOR.OUT
INCEPUT	3
ATRIB	
ALEGE	
CAZ	
ATRIB	
CAZ	
ATRIB	
CAZ_IMPLICIT	
ATRIB	
SFARSIT	

Observații

- se execută ATRIB de la primul CAZ;
- se execută ATRIB de la al doilea CAZ;
- se execută ATRIB de la CAZ_IMPLICIT.

Exemplul 5

ALGOR.IN	ALGOR.OUT
INCEPUT	4
atrib	
DACA	
ATUNCI	
ATRIB	
ALTFEL	
ATRIB	
pentru	
atrib	
SFARSIT	

Observații

- se execută ATUNCI și PENTRU;
- se execută ATUNCI și nu se execută PENTRU;
- se execută ALTFEL și PENTRU;
- se execută ALTFEL și nu se execută PENTRU.

Exemplul 6

ALGOR.IN	ALGOR.OUT
INCEPUT	6
PENTRU	
INCEPUT	
ATRIB	
daca	
atunci	
atrib	
altfel	
INCEPUT	
DACA	
ATUNCI	
ATRIB	
DACA	
ATUNCI	
ATRIB	
ALTFEL	
ATRIB	
SFARSIT	
SFARSIT	
SFARSIT	

Timp maxim de execuție

1 secundă/test

P050002: Cod

prof. Eugen Ionescu, Liceul "Tiberiu Popoviciu", Cluj-Napoca

Pentru a concura cu numărul de serie de la procesoarele *Intel Pentium III*, *AMD* a stabilit un sistem de identificare pentru noile procesoare cu numele de cod *Thunderbird*. Fiecare firmă distribuitoare primește o mulțime de litere (de exemplu: {a,m,x}) din care va trebui să-și formeze codurile proprii de identificare. Firmelor li se impune exact de câte ori trebuie să apară fiecare literă în aceste coduri. De exemplu, o firmă trebuie să formeze identificatori care să conțină exact trei litere a, două litere m și o literă x.

Cerință

Scrieți un program care, cunoscând un anumit cod dat, determină următorul cod corect în ordine lexicografică, dacă există un astfel de cod următor.

Date de intrare
Singura linie a fișierului de intrare **COD.IN** conține un cod. Codurile sunt formate din cel mult 100 de litere mici ale alfabetului latin.

Date de ieșire
Fișierul de ieșire **COD.OUT** va conține o singură linie pe care se va afla codul următor; dacă nu există un astfel de cod, atunci în fișier se va scrie `Este ultimul cod.`

Exemplul 1

COD.IN	COD.OUT
amaaxm	amamax

Exemplul 2

COD.IN	COD.OUT
xmmaaa	Este ultimul cod.

Timp maxim de execuție
1 secundă/test

P050003: Comoara
Mihai Stroe, student, Universitatea Politehnică, București

Cei K membri ai unui grup de căutători de comori se află într-un complex dreptunghiular format din camere pătrate de latură 1. În mod normal toate camerele ar trebui să fie deschise, dar o parte dintre ele sunt închise și pot fi deschise doar cu ajutorul unor cartele speciale. Astfel, fiecare cameră și fiecare cartelă are asociat un număr între 0 și 20; o cameră cu numărul asociat 0 este deschisă de la început, în timp ce una cu numărul asociat diferit de 0 este inițial închisă, poate fi deschisă din interior sau din exterior dintr-o cameră vecină cu o cartelă cu numărul corespunzător și va rămâne deschisă ulterior. Cartelele cu numărul asociat 0 nu au nici o întrebuințare practică. Un număr poate fi asociat mai multor cartele, respectiv camere.

Dintr-o cameră se poate trece în alta numai dacă ele sunt vecine și ambele sunt deschise. Două camere se consideră vecine dacă au un perete (deci o latură) în comun.

În fiecare cameră se află comori cu o valoare între 0 și 10000. De asemenea, în fiecare cameră se află exact o cartelă de acces (cu numărul asociat între 0 și 20); un căutător de comori poate ridica și folosi cartelele din camerele prin care trece, dar nu le poate da unui alt căutător decât dacă respectivul se află în aceeași cameră cu el.

Cerință
Cunoscându-se configurația complexului și pozițiile inițiale ale căutătorilor de comori, să se determine valoarea maximă a comorilor adunate de aceștia.

Date de intrare
Datele de intrare se citesc din fișierul **COMOARA.IN** care conține:
- pe prima linie dimensiunile complexului, m și n;
- pe următoarele m linii numerele asociate fiecărei camere;
- pe următoarele m linii valorile comorilor din fiecare cameră;
- pe următoarele m linii numerele asociate cartelelor din fiecare cameră;
- pe următoarea linie numărul K al membrilor grupului;
- pe următoarele K linii pozițiile inițiale ale membrilor grupului în complex.

Date de ieșire
În fișierul **COMOARA.OUT** se va afișa valoarea totală a comorilor care pot fi strânse de cei K membri ai grupului.

Restricții
- $m,n \leq 40$
- $k \leq 10$

Exemplu

COMOARA.IN	COMOARA.OUT
3 3 0 0 11 14 0 10 19 0 0 5162 4331 1390 5230 1955 9796 5507 6210 1021 0 0 0 19 0 0 0 0 0 2 3 2 2 1	23909

Observație
Al doilea căutător de comori nu poate pătrunde în camera (3,1), deși are cartela corespunzătoare, deoarece nu poate părăsi camera (2,1) în care este plasat inițial.

Timp maxim de execuție
1 secundă/test

P050004: Cuburi
prof. Ovidiu Domșa, Colegiul "Horea, Cloșca și Crișan", Alba Iulia

Un joc nou este format din n cuburi identice, numerotate de la 1 la n. Fiecare cub are toate cele șase fețe adezive (cu "scai"). Jocul constă în realizarea unui obiect din toate cele n cuburi. Obiectul inițial este cubul 1. Un obiect se obține din obiectul anterior prin alipirea unui nou cub la un cub aflat în obiect, pe una din fețele acestuia. Pentru alipirea unui cub nou, se extrage un bilețel cu numărul cubului la care se va alipi acesta și se aruncă un zar care va indica unde va fi lipit cubul nou (sus, jos, la stânga, la dreapta, în față sau în spate, poziții codificate respectiv cu 1, 2, 3, 4, 5, 6, ale cubului din obiect.

Cerințe

a) Dându-se o succesiune de alipire a cuburilor, verificați dacă aceasta este validă

b) Dacă succesiunea este validă, precizați dacă obiectul obținut are forma unui paralelipiped plin, specificând dimesiunile acestuia.

Date de intrare

Fișierul `CUBURI.IN` conține pe prima linie valoarea lui `n`. Pe următoarele `n-1` linii, se află o succesiune de așezare a cuburilor, pentru fiecare cub specificând:
număr_cub bilețel zar, valori separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire `CUBURI.OUT` va conține pe două linii rezultatele de la punctul a) și respectiv b) după cum urmează:

a) Dacă succesiunea este validă, prima linie va conține valoarea 0.
În caz contrar, prima linie va conține numărul cubului ce nu poate fi alipit, numărul cubului și numărul feței la care nu se poate alipi, precum și codul de eroare:
1) pentru alipire la cub inexistent;
sau
2) pentru alipire pe o față ocupată.

b) Dacă obiectul nu este paralelipiped plin, sau dacă succesiunea nu este validă, a doua linie a fișierului va conține valoarea 0.
Dacă obiectul este paralelipiped plin, a doua linie a fișierului va conține dimensiunile acestuia, măsurate în număr de cuburi, în ordinea:
număr de cuburi pe direcțiile sus-jos, stânga-dreapta, față-spate, separate prin câte un spațiu.

Restricții

- $2 \leq n \leq 1000$
- pe o dimensiune se alipesc cel mult 10 cuburi.

Exemplul 1

CUBURI.IN	CUBURI.OUT
6	0
2 1 1	3 2 1
3 2 4	
5 1 4	
6 3 1	
4 6 3	

Exemplul 2

CUBURI.IN	CUBURI.OUT
8	6 3 6 1
2 1 1	0
5 1 4	
6 3 6	
7 6 2	
3 2 4	
4 2 6	
8 7 3	

Exemplul 3

CUBURI.IN	CUBURI.OUT
8	8 7 3 2
2 1 1	0
4 2 6	
3 2 4	
6 3 6	
7 6 2	
5 1 6	
8 7 3	

Exemplul 4

CUBURI.IN	CUBURI.OUT
4	0
2 1 1	0
4 2 1	
3 2 4	

Timp maxim de execuție

1 secundă/test

P050005: Fibo

prof. Marinel Șerban, Liceul de Informatică, Iași

Considerăm șirul lui Fibonacci: `0,1,1,2,3,5,8,13,21,...` Fiind dat un număr natural ($n \in \mathbb{N}$), scrieți acest număr sub formă de sumă de elemente neconsecutive din șirul Fibonacci, fiecare element putând să apară cel mult o dată, astfel încât numărul de termeni ai sumei să fie minim.

Date de intrare

Din fișierul de intrare `FIB.IN` se citește de pe prima linie numărul natural `n`. Acesta poate avea cel mult 80 de cifre.

Date de ieșire

În fișierul de ieșire `FIB.OUT` se vor afișa termeni ai șirului Fibonacci, câte unul pe linie, a căror sumă este `n`.

Exemplul 1

FIB.IN	FIB.OUT
20	2
	5
	13

Exemplul 2

FIB.IN	FIB.OUT
8	8

Timp maxim de execuție

1 secundă/test

P050006: Kommando

prof. Marinel Șerban, Liceul de Informatică, Iași

Într-o clădire cu `h` etaje sunt reținuți, la parter, câțiva prizonieri de către `T` teroriști. Fiecare etaj al clădirii are `m×n`

camere identice. Fiecare cameră are un cod numeric (nu neapărat unic) exprimat printr-un număr din intervalul 0-255. O trupă de kommando, formată din K specialiști în lupte antitero, trebuie să elibereze prizonierii. Trupa de kommando este parașutată pe clădire și încearcă să ajungă la parter.

Se cunoaște locul (x,y) unde fiecare membru al trupei a aterizat pe acoperiș. Greutatea fiecărui membru al trupei este exprimată în unități din intervalul 1-255. Un membru al trupei poate trece "în jos" printr-o cameră a clădirii doar dacă "greutatea lui trece prin camera respectivă", conform următoarei definiții.

Definiție

Spunem că "a *trece prin* b" (a>>b) dacă, în reprezentare binară, numărul de cifre 1 a lui a este mai mic sau egal cu numărul de cifre 1 a lui b și cifrele 1 ale lui a sunt comune cu unele cifre 1 ale lui b.

Exemplu

"44 *trece prin* 174" (44>>174) deoarece

44 = 00101100

174 = 10101110

Pentru detectarea unei camere prin care să poată trece, un membru al trupei de kommando poate să se deplseze cu un "pas" în cele 8 direcții alăturate poziției curente în care a ajuns prin aterizare sau trecând "în jos". Prin "pas"-ul respectiv se ajunge la una din cele 8 camere vecine. Prizonierii pot fi eliberați doar dacă la parter ajung cel puțin T membri ai trupei de kommando. Să se determine dacă prizonierii pot fi eliberați sau nu, precum și numărul de membri ai trupei de kommando care pot să ajungă la parter.

Date de intrare

Fișierul text **KOMMANDO.IN** are structura următoare:

- pe prima linie sunt scrise valorile m, n, h, K, T despărțite prin câte un spațiu, cu semnificațiile descrise mai sus;
- următoarele h linii reprezintă codurile celor m×n camere ale unui etaj, despărțite prin câte un spațiu;
- ultimele K linii ale fișierului conțin greutatea și coordonatele x și y ale pozițiilor de aterizare pe acoperiș ale celor K membri ai trupei de kommando, pentru fiecare câte o linie:

m n h K T

c_{111} c_{112} ... c_{11n} c_{121} c_{122} ... c_{12n} ... c_{1m1} c_{1m2} ... c_{1mn}

c_{211} c_{212} ... c_{21n} c_{221} c_{222} ... c_{22n} ... c_{2m1} c_{2m2} ... c_{2mn}

...

c_{h11} c_{h12} ... c_{h1n} c_{h21} c_{h22} ... c_{h2n} ... c_{hm1} c_{hm2} ... c_{hmn}

G_1 x_1 y_1

G_2 x_2 y_2

...

G_K x_K y_K

Date de ieșire

În fișierul text **KOMMANDO.OUT** se va scrie:

- pe prima linie DA sau NU;
- pe linia a doua - numărul de membri ai trupei de kommando ajunși la parter.

Restricții

- $2 \le m,n,h \le 35$, $1 \le x_i \le m$, $1 \le y_i \le n$
- $1 \le T, K, G_i \le 255$
- $0 \le c_{ijk} \le 255$
- Toate valorile sunt întregi.

Exemplu

KOMMANDO.IN

```
5 5 5 3 2
0 0 0 0 0 0  0 33  0 0 0  0 2  0 0 0 0 0 3 0  0 0 0 0 0
0 0 0 0 0 0  1  0  0 0 0 44 2  0 0 0 0 0 3 0  0 0 0 0 0
0 0 0 0 0 11 0  0  0 0 0  0 2 22 0 0 0 0 0 3  0 0 0 0 0
0 0 0 0 0 0  0  0  0 0 0 66 2  0 0 0 0 0 7 0 15 0 0 0 0
0 0 0 0 0 0  0  0  0 0 0  0 0  0 0 0 0 0 0 0  0 0 0 0 0
1 2 2
2 3 3
3 4 4
```

KOMMANDO.OUT

```
DA
2
```

Timp maxim de execuție

5 secunde/test

Clasa a X-a

P050007: Pariul broscuțelor

prof. Stelian Ciurea, Universitatea "Lucian Blaga", Sibiu

Câteva broscuțe din orașul C..., mergând la plimbare prin parcul din centrul orașului și descoperind un mozaic asemănător unei table de șah de dimensiune n×n, au inventat un joc cu următoarele reguli:

- fiecare broscuță pornește din centrul pătrățelului aflat în colțul din stânga-sus al tablei, trece prin fiecare pătrățel al mozaicului, și se întoarce în pătrățelul de unde a pornit;
- o broscuță poate sări din centrul unui pătrățel oarecare în centrul oricăruia din pătrățelele vecine aflate în direcțiile: N, S, E, V, NE, NV, SE, SV;
- dacă se unesc centrele pătrățelelor în ordinea în care au fost parcurse, se obține o linie închisă care nu trebuie să se autointersecteze.

Una dintre broscuțe - mai nechibzuită - face pariu pe greutatea ei în musculițe, că poate găsi o ordine în care să acopere cu sărituri mozaicul, astfel încât lungimea drumului parcurs să nu poată fi depășită.

Lungimea este în sens geometric: drumul dintre centrele a două pătrate cu o latură comună este 1, iar între centrele a două pătrate ce au doar un vârf comun, distanța este $\sqrt{2}$.

Cerință

Din păcate, broscuța nu prea știe cum să procedeze, așa că vă roagă să îi indicați ordinea în care trebuie să parcurgă păt. ățelele tablei, astfel încât să nu piardă pariul.

Date de intrare

Fișierul de intrare **FROG.IN**, conține pe prima linie valoarea lui n.

Date de ieșire

În fișierul **FROG.OUT** se vor scrie, pe linia i (i=1...n^2+1), două perechi de valori separate printr-un spațiu, reprezentând coordonatele pătrățelului unde trebuie să sară broscuța la pasul i (se consideră că în colțul din stânga sus avem coordonatele 1,1, iar localizarea unui pătrățel se face ca în matrice).

Restricție

$2 \le n \le 250$

Exemplu

FROG.IN	FROG.OUT poate conține:
4	1 1
	2 2
	2 1
	3 1
	3 2
	4 1
	4 2
	3 3
	4 3
	4 4
	3 4
	2 3
	2 4
	1 4
	1 3
	1 2
	1 1

Timp maxim de execuție

3 secunde/test

P050008: Castel

prof. Dana Lica, Liceul "Ion Luca Caragiale", Ploiești

Unui muncitor i se încredințează o misiune grea. El are de lăcuit toate coridoarele unui castel. Pentru aceasta el primește o hartă în care sunt reprezentate coridoarele și cele N puncte de intersecție ale acestora (numerotate de la 1 la N). Unele puncte de intersecție sunt considerate de muncitor ca fiind "puncte speciale" deoarece ele sunt singurele puncte care îi permit ieșirea și intrarea în castel, prin intermediul unor uși.

Deoarece operația de lăcuire trebuie să fie efectuată în cel mai scurt timp, muncitorului i s-a impus ca orice coridor să nu fie traversat decât o singură dată, adică doar atunci când este lăcuit.

În aceste condiții, muncitorul trebuie să lăcuiască toate coridoarele, folosindu-se eventual de "punctele speciale" ale castelului, plecând din orice intersecție de coridoare dorește și terminând operația în același punct.

Cerință

Determinați un traseu prin care operația este satisfăcută.

Date de intrare

Fișierul de intrare **CASTEL.IN** conține:

- Pe prima linie numărul N.
- Pe a doua linie numărul de coridoare M.
- Pe următoarele M linii perechi de două numere reprezentând perechi de puncte între care există coridor care trebuie lăcuit.
- Pe următoarea linie numărul P reprezentând numărul de "puncte speciale".
- Pe ultima linie șirul "punctelor speciale" separate prin câte un spațiu.

Date de ieșire

Ieșirea se va face în fișierul **CASTEL.OUT** care va avea formatul următor:

- Pe prima linie se va scrie mesajul DA sau NU, după cum există sau nu un traseu ciclic care respectă cerințele.
- În caz afirmativ fișierul va conține pe a doua linie un șir de numere reprezentând punctele în ordinea de apariție pe traseu, despărțite după caz de secvența de caractere ' - ' (*spațiu minus spațiu*) sau ' * ' (*spațiu asterisc spațiu*).
- O secvență de tipul i - j semnifică faptul că muncitorul a traversat și lăcuit coridorul dintre punctele i și j. O secvență de tipul i * j semnifică faptul că muncitorul a părăsit castelul prin "punctul special" i și a revenit în castel prin "punctul special" j.

Restricții

- $1 \le N \le 5000$
- $0 \le M \le 10000$

Exemplu

CASTEL.IN	CASTEL.OUT poate conține:
6	DA
6	1 - 2 - 5 * 6 - 4 - 3 - 2 * 4 - 1
1 2	
2 3	
3 4	
4 1	
2 5	
4 6	
5	
1 2 5 4 6	

Notă
- Jumătate din testele folosite pentru evaluare vor avea dimensiunea $N \leq 100$
- În cazul în care nici un test cu soluție nu va fi trecut, punctajele pentru testele pentru care răspunsul este NU nu vor fi acordate.

Timp maxim de execuție
2 secunde/test

P050009: Drumuri scurte

prof. Stelian Ciurea, Universitatea "Lucian Blaga", Sibiu

Se consideră un graf turneu cu n vârfuri. Numim graf turneu un graf orientat la care între oricare două vârfuri se află un arc și numai unul.

Cerință
Să se găsească o dispunere a arcelor grafului astfel încât între oricare două noduri să existe un drum de lungime 1 sau 2.

Date de intrare
Fișierul de intrare **GRAF.IN**, conține pe prima linie valoarea lui n.

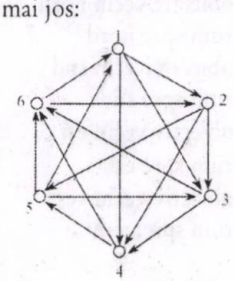
Date de ieșire
Ieșirea se va face în fișierul **GRAF.OUT** care va conține:
- matricea de adiacență a grafului care îndeplinește cerințele problemei (pe linia i a fișierului se va afla linia i a matricei) în cazul în care există soluție;
- textul `fara solutie` (cu litere mici!), dacă nu există nici o posibilitate de dispunere a arcelor.

Restricție
$2 \leq n \leq 250$

Exemplu

GRAF.IN	**GRAF.OUT** poate conține:
6	0 1 1 1 0 0
	0 0 1 1 1 0
	0 0 0 1 0 1
	0 0 0 0 1 1
	1 0 1 0 0 1
	1 1 0 0 0 0

Explicație
Matricea de adiacență din **GRAF.OUT** corespunde grafului din imaginea de mai jos:



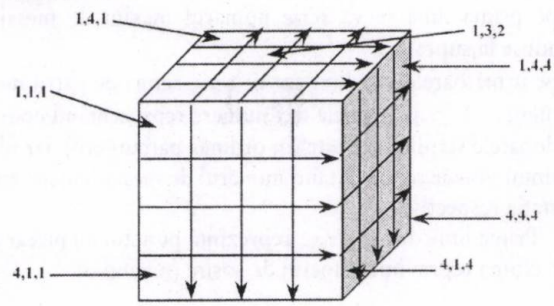
Timp maxim de execuție
3 secunde/test

P050010: Lac

prof. Dan Grigoriu, Colegiul Național "Tudor Vianu", București

Un lac are forma cubică, cu latura M (număr natural). El este împărțit în M^3 poziții elementare. O poziție (o căsuță) este identificată prin coordonatele n, l, c, ca într-un spațiu tridimensional, unde:
- n reprezintă nivelul (1 pentru cel din imediata apropiere a suprafeței apei, M pentru nivelul de pe fundul lacului);
- l reprezintă linia (linia 1 este linia cea mai apropiată de privitor);
- c reprezintă coloana (coloana 1 este cea din stânga privitorului).

Exemplu
Pentru m=4



Exemple de deplasări elementare:
- din poziția (4, 2, 2) se poate ajunge la una din pozițiile: (3, 1, 1), (3, 1, 2), (3, 1, 3), (3, 2, 1), (3, 2, 2), (3, 2, 3), (3, 3, 1), (3, 3, 2), (3, 3, 3);
- din poziția (2 ,1, 1) se poate ajunge la una din pozițiile: (1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2).

La suprafața apei, considerată pe nivelul 0, există o căsuță specială, și anume cea în care trebuie să ajungă în final un scafandru ce pleacă de pe fundul lacului.

În lac există poziții elementare considerate ca fiind stații de lucru ce au de transmis la suprafață anumite mesaje. Aceste stații pot fi oricâte și pot fi dispuse oriunde în cub (chiar și în poziția de plecare a scafandrului).

Scafandrul se poate deplasa din poziția sa numai într-o poziție de pe nivelul imediat superior (de la nivelul i la nivelul i-1, pentru $1 \leq i \leq m$) și care diferă ca linie și coloană cu cel mult 1 de poziția precedentă. Deci dintr-o poziție se poate ajunge în cel mult 9 poziții pe nivelul imediat superior (vezi figura).

Scafandrul se află pe fundul lacului la coordonatele m, l_1, c_1. El trebuie să ajungă la suprafața lacului (adică deasupra primului nivel al cubului), într-o poziție de coordonate 0, l_2, c_2, adică imediat deasupra poziției 1, l_2, c_2 din cub.

Cerință

Trebuie determinat drumul pe care trebuie să-l aleagă scafandrul pentru a duce la destinație cât mai multe mesaje, respectând restricțiile din enunț.

Date de intrare

Fișierul de intrare **LAC.IN**, conține:

- pe prima linie cinci numere `m` l_1 c_1 l_2 c_2 despărțite prin câte un spațiu, `m` reprezentând dimensiunea cubului, l_1, c_1 linia și coloana poziției de plecare, iar l_2, c_2 linia și coloana poziției de sosire;
- pe a doua linie numărul `y` de stații din lac;
- pe următoarele `y` linii seturi de câte patru numere `n l c p`. Primele trei numere `n l c` reprezintă coordonatele unei stații, iar ultimul număr (`p`), reprezintă numărul de mesaje ce trebuie transmise de stația respectivă.

Date de ieșire

Ieșirea se va face în fișierul **LAC.OUT** în formatul următor:

- pe prima linie se va scrie numărul maxim de mesaje aduse la suprafață.
- pe următoarele `m+1` linii se vor scrie seturi de patru numere `n l c p`, primele trei numere reprezentând coordonatele stațiilor vizitate (în ordinea parcurgerii), iar ultimul număr reprezentând numărul de mesaje culese din stația respectivă.

Prima linie din cele `m+1` reprezintă punctul de plecare, iar ultima repreyintă punctul de sosire (nivelul `0`).

Restricții

- $2 \leq M \leq 20$
- $0 \leq P \leq 100$

Exemplu

LAC.IN

```
3 2 2 2 2
6
1 3 1 10
2 1 3 9
2 2 1 6
2 3 2 7
2 3 3 8
3 2 2 5
```

LAC.OUT poate conține:

```
22
3 2 2 5
2 3 2 7
1 3 1 10
0 2 2 0
```

Timp maxim de execuție

1 secundă/test

Note

- Nu este permisă ieșirea din lac decât prin punctul de sosire.
- În punctul de sosire nu se află nici o stație.
- Dacă pentru o anumită poziție toate mutările posibile sunt în căsuțe fără stații, atunci în fișierul de ieșire se va scrie o linie cu coordonatele căsuței alese și valoarea `0` pentru numărul de mesaje (ca pentru o stație cu `0` mesaje).

P050011: Obiective turistice

Cristian Cadar, student, Universitatea Politehnică, București

În județul Constanța se găsesc cel mult `2500` de obiective turistice dispuse pe o suprafață asemănătoare cu un caroiaj format din pătrățele având latura de `1` kilometru. Obiectivele sunt amplasate în unele puncte de intersecție ale caroiajului.

Două obiective se consideră vecine dacă sunt situate pe una din direcțiile `Nord`, `Sud`, `Est`, `Vest` la o distanță de `1` kilometru una față de alta.

Între anumite perechi de obiective vecine au fost construite drumuri directe, pentru a se asigura deplasarea între oricare două obiective.

Despre fiecare obiectiv avem următoarele informații:

- numărul asociat acestuia, care este unic și are o valoare cuprinsă între `1` și numărul total al obiectivelor;
- obiectivele vecine cu care acesta este conectat prin drumuri directe (cel mult patru obiective).

Pentru fiecare două obiective care comunică se cunoaște costul asfaltării drumului direct dintre ele.

Cerințe

1. Se dorește realizarea unei hărți a județului în care să apară reprezentate toate obiectivele. Harta se va reprezenta sub forma unei matrice având dimensiunea maximă `50x50`. Fiecare element al matricei codifică printr-o succesiune de patru cifre binare existența drumurilor directe care încadrează un pătrățel din caroiaj. Dacă pe latura din nord a acestui pătrățel există un drum direct între două obiective, atunci bitul cel mai semnificativ va fi `1`. În caz contrar acesta va fi `0`. Ceilalți trei biți corespund, în ordine, direcțiilor `Sud`, `Est`, `Vest`. De exemplu, dacă pentru un pătrățel există drumuri pe laturile `Nord`, `Sud` și `Vest`, cei patru biți vor fi: `1 1 0 1`. Harta trebuie să aibă suprafață minimă dar să cuprindă toate obiectivele.
2. Se dorește asfaltarea unora dintre drumurile existente astfel încât să se poată ajunge, mergând numai pe drumuri asfaltate de la oricare obiectiv la oricare altul. Costul total al asfaltării trebuie să fie minim. Să se precizeze care dintre drumuri vor fi asfaltate.

Date de intrare

Fișierul de intrare **OB.IN** are un număr necunoscut de linii, fiecare fiind de forma:

```
NO nvN cdsN nvS cdsS nvE cdsE nvV cdsV
```

având semnificația:

`NO` = număr obiectiv
`nvN` = număr obiectiv vecin nord
`cdsN` = cost drum spre nord
`nvS` = număr obiectiv vecin sud
`cdsS` = cost drum spre sud
`nvE` = număr obiectiv vecin est
`cdsE` = cost drum spre est
`nvV` = număr obiectiv vecin vest
`cdsV` = cost drum spre vest

Dacă un anumit obiectiv nu are vecin într-o anumită direcție, numărul vecinului din direcția respectivă va fi înlocuit cu 0, iar costul drumului direct dintre cele două cu 0.

Date de ieșire

Datele de ieșire se scriu în fișierele **OB1.OUT** și **OB2.OUT**.

- Fișierul **OB1.OUT** conține o hartă a obiectivelor folosind numere între 0 și 15. Fiecare număr reprezintă codificarea în baza zece a celor patru cifre binare.
- Fișierul **OB2.OUT** conține pe prima linie costul optim cerut și pe următoarele linii obiectivele între care se asfaltează drumul.

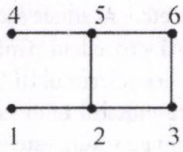
Restricții

- numărul de obiective este cel mult 2500;
- harta se încadrează într-o matrice de 50 de linii și 50 de coloane;
- lungimile drumurilor directe sunt cel mult 100.

Exemplu

OB.IN	OB1.OUT	OB2.OUT
1 0 0 0 0 2 3 0 0	1 2	11
3 6 3 0 0 0 0 2 1	14 15	1 2
6 0 0 3 3 0 0 5 2		2 3
2 5 2 0 0 3 1 1 3		6 5
4 0 0 0 0 5 3 0 0		2 5
5 0 0 2 2 6 2 4 3		5 4

Fișierului **OB1.OUT** îi corespunde următoarea hartă:



Note

- Drumurile directe între obiective sunt bidirecționale.
- Se garantează că datele de intrare nu conțin date incompatibile.
- Se garantează că nici una din dimensiunile hărții nu va depăși 50.
- Punctajele pentru cele două cerințe sunt independente.

Timp maxim de execuție

2 secunde/test

P050012: Călătorie cu taxiul

Cristian Cadar, student, Universitatea Politehnică, București

Un călător dorește să străbată o distanță de n kilometri cu taxiul, astfel încât prețul pe care îl plătește să fie minim. Compania de taximetre pe care această persoană o folosește are un tarif unitar fix de C lei pentru un kilometru, însă are de asemenea câteva oferte speciale de tipul (k,L) cu semnificația că pentru o porțiune de drum de k kilometri călătorul poate să plătească L lei (în locul unui tarif unitar pe numărul de kilometri). Din păcate, aceste oferte sunt foarte ciudate, și dacă nu este atent, călătorul nostru se poate păcăli ușor. De exemplu compania de taximetre poate să ceară pentru 2,43 km un tarif de 1370,89 lei și pentru 3,8 km un tarif de 1509,43 lei. Cum călătorul nostru are dificultăți în manevrarea unor astfel de numere, el vă roagă să-l ajutați să afle modul optim de organizare a excursiei sale, cunoscând atât numărul de kilometri pe care acesta trebuie să-i parcurgă cât și tarifele practicate de compania de taximetre.

Date de intrare

- Pe prima linie a fișierului **TAXI.IN** se găsește numărul n ($n \le 100$) de kilometri pe care călătorul trebuie să îi parcurgă.
- Pe cea de-a doua linie se află tariful unitar L practicat de compania de taximetre.
- Următoarele linii (cel mult 100), până la sfârșitul fișierului conțin perechi de forma k C reprezentând ofertele speciale ale companiei. Toate numerele care apar în fișierul de intrare sunt numere reale, strict pozitive, mai mici decât 1000 și cu exact două zecimale.

Date de ieșire

- Prima linie a fișierului **TAXI.OUT** va conține suma minimă pe care o poate achita călătorul, scrisă cu trei zecimale exacte.
- Următoarea linie va conține o succesiune de numere, reprezentând ordinea în care se face alegerea ofertelor speciale. Aceste numere pot fi atât naturale cât și numere reale negative. Numerele naturale indică a câta ofertă din fișierul de intrare a fost aleasă la un moment dat, iar numerele reale negative indică numărul de kilometri parcurși (în modul), folosind prețul unitar.

Restricții

- $0 < n \le 100$
- $0 < L \le 1000$
- $0 < k \le 1000$
- $0 < C \le 1000$
- Numerele n, L, k, C sunt reale și au maximum câte două zecimale.

Exemplu

TAXI.IN

```
52.65           // nr. km
0.80            // tarif unitar
8.75 5.50       // 8.75 km cu prețul de 5.5
60.35 20.47     // 60.35 km cu prețul de 20.47
20.40 10.20     // 20.4 km cu prețul de 10.2
```

TAXI.OUT poate conține:

```
28.380
-0.69 1 3 3 -2.41
```

Timp maxim de execuție

2 secunde/test

P050013: Arbore

Marius Vlad, student, Universitatea Politehnică, București

Se dă un arbore format din n noduri în care fiecare nod are asociat un număr natural nenul.

Cerință

Să se selecteze din arborele inițial un **subgraf conex** pentru care suma valorilor asociate nodurilor este egală cu un număr natural k dat.

Un subgraf este graful din care se elimină noduri împreună cu muchiile aferente.

Date de intrare

Nodurile arborelui sunt numerotate de la 1 la n, rădăcina fiind nodul numerotat cu 1.

Fișierul de intrare **ARBORE.IN**, are următoarea structură:

- pe prima linie sunt scrise numerele n și k;
- pe cea de-a doua linie este scrisă valoarea asociată nodului 1 (rădăcina arborelui);
- pe cea de-a treia linie sunt scrise două numere, primul reprezentând nodul părinte al nodului 2, iar al doilea reprezentând valoarea asociată nodului 2;
- pe cea de-a patra linie sunt scrise două numere, primul reprezentând nodul părinte al nodului 3, iar al doilea reprezentând valoarea asociată nodului 3;

...

- pe linia n+1 sunt scrise două numere, primul reprezentând nodul părinte al nodului n, iar al doilea reprezentând valoarea asociată nodului n.

Date de ieșire

Ieșirea se va face în fișierul **ARBORE.OUT**:

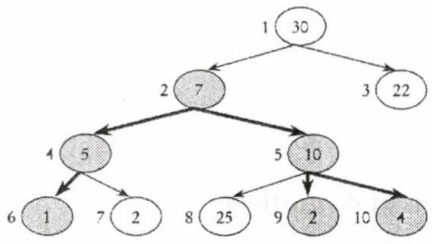
- dacă nu există soluție se va afișa doar numărul -1 pe prima linie;
- dacă există soluție se vor afișa nodurile ce formează un subgraf conex care respectă cerința problemei (câte un singur nod pe fiecare linie).

Restricții

- $1 \le n \le 100$
- $1 \le k \le 1000$
- $1 \le$ *valoarea asociată unui nod* ≤ 1000

Exemplu

Pentru arborele:



ARBORE.IN	**ARBORE.OUT** poate conține:
10 29	2
30	4
1 7	5
1 22	6
2 5	9
2 10	10
4 1	
4 2	
5 25	
5 2	
5 4	

Timp maxim de execuție:

3 secunde/test

P050014: Moara

prof. Ion Maxim, Inspectoratul Școlar Județean, Suceava

Motto: Una-i să sortezi cartoane,
alta-i să sortezi vagoane!

La poarta unei mori se află n saci, etichetați cu numere de la 1 la n, așezați în linie, într-o ordine dată. Sacul cu eticheta i are greutatea g_i.

Morarul cere ucenicului său să rearanjeze sacii în ordinea crescătoare a etichetelor. Cum nu este spațiu de manevră pentru a schimba sacii între ei, ucenicul ia un scaun și-l așează lângă poziția k din șir, alege un sac și-l pune pe scaun, ia alt sac și-l mută în locul gol, apoi aduce alt sac în locul eliberat de acesta etc. sau aduce sacul de pe scaun în locul rămas gol. Repetă procedeul până când sacii ajung în ordinea cerută de morar. Scaunul nu se mută. Efortul depus de ucenic pentru mutatul unui sac, indiferent dacă acesta a fost pus sau nu pe scaun, este egal cu produsul dintre greutatea sacului și distanța pe care a fost transportat sacul. Distanța dintre doi saci alăturați este egală cu unitatea.

Cerință

Se cere să se stabilească poziția amplasării scaunului și ordinea schimbării sacilor între ei, astfel încât ucenicul să facă un număr minim de mutări și efortul depus de acesta să fie minim.

Date de intrare

Fișierul de intrare **MOARA.IN** conține pe prima linie numărul de saci n și pe liniile următoare, ordinea așezării sacilor, urmată de șirul greutăților sacilor $g_1, g_2, ..., g_n$, unde g_i este greutatea sacului cu eticheta i.

Date de ieșire

Fișierul **MOARA.OUT** va conține pe prima linie numerele p k e, unde p este poziția scaunului, k numărul de mutări și e valoarea efortului depus. Următoarele linii vor conține câte o mutare de forma d s, unde d este poziția destinație și s este poziția sursă, poziția scaunului se va nota cu 0 (zero!).

Restricții

- $2 \le n \le 10000$
- $1 \le k \le n$
- $1 \le g_i \le 255$, pentru $1 \le i \le n$

Exemplu

MOARA.IN	MOARA.OUT
5	3 5 25
2 4 3	0 2
5 1	2 1
3 5 1 2 4	1 5
	5 4
	4 0 (soluția nu este unică!)

Timp maxim de execuție

1 secundă/test

P050015: Puncte

prof. Doru Anastasiu Popescu, Liceul "Radu Greceanu", Slatina

Se dau n puncte situate pe un cerc P_1, P_2, ..., P_n și un număr natural k. Se cere să se verifice dacă se pot construi k segmente cu ambele capete în mulțimea formată din punctele P_1, P_2, ..., P_n, astfel încât să nu se formeze nici un triunghi cu vârfurile în aceste puncte. Dacă există mai multe soluții se cere una dintre ele.

Date de intrare

Fișierul de intrare **PUNCTE.IN** conține pe prima linie numărul n, iar pe a doua linie numărul k.

Date de ieșire

Fișierul **PUNCTE.OUT** va conține:

- o linie pe care se află caracterul 0, dacă nu există soluție;
- k+1 linii, dintre care, pe prima linie se află caracterul 1 (având semnificația că există soluție), iar pe umătoarele k linii se află câte două numere separate printr-un spațiu (de forma i j având semnificația că P_iP_j reprezintă un segment), în cazul când există soluție.

Restricții

- $1 \le n \le 300$
- $1 \le k \le 300$

Exemplul 1

PUNCTE.IN	PUNCTE.OUT
4	0
5	

Exemplul 2

PUNCTE.IN	PUNCTE.OUT poate conține:
4	1
2	1 4
	1 2

Timp maxim de execuție

1 secundă/test

P050016: SICN

prof. Emanuela Cerchez, Liceul de Informatică, Iași

Serviciul de Informații al Comisiei Naționale (*SICN*) este constituit din n agenți ($1 \le n \le 150$), având numere de cod distincte de la 1 la n și un agent șef codificat cu 0.

Din motive de securitate, nu oricare doi agenți au contacte (informaționale!) directe, dar prin contactele directe existente, oricare doi agenți își pot comunica informații.

Un serviciu este considerat *forte* dacă și numai dacă nu conține nici un agent prin suprimarea căruia se compromite comunicarea (și ca urmare, să existe agenți care să nu își mai poată transmite informații).

Cerințe

Scrieți un program, care verifică dacă *SICN* este forte. În plus, dacă *SICN* nu este forte, programul trebuie să determine:

- verigile slabe ale *SICN* (agenții prin a căror suprimare se compromite comunicarea);
- toate grupurile de agenți care sunt forte în cadrul *SICN*, maximale cu această proprietate.

Date de intrare

Fișierul de intrare **SICN.IN** conține pe prima linie numărul de agenți n, din *SICN*, (exclusiv șeful), iar pe fiecare din următoarele linii câte o pereche de numere x y unde $x, y \in \{0, 1, 2, ..., n\}$, sunt separate printr-un spațiu, și au semnificația *x și y au contact direct.*

Date de ieșire

Fișierul de ieșire, **SICN.OUT**, va conține mesajul SICN este forte sau:

- pe prima linie, mesajul SICN nu este forte;
- pe a doua linie, numerele de cod ale agenților care constituie verigile slabe ale serviciului, în ordine crescătoare, separate prin spații;
- pe a treia linie, m - numărul de grupuri forte în cadrul *SICN*;
- pe fiecare din următoarele m linii, numerele de cod ale agenților fiecărui grup forte, în ordine crescătoare, separate prin spații.

Exemplul 1

SICN.IN	SICN.OUT
2	SICN este forte
0 2	
0 1	
2 1	

Exemplul 2

SICN.IN	SICN.OUT
3	SICN nu este forte
0 1	0 1
0 3	3
2 1	0 3
	1 2
	0 1

Timp maxim de execuție
1 secundă/test

P050017: Spioni

prof. Rodica Pintea, Liceul "Grigore Moisil", București

O rețea formată din n spioni, numerotați de la 1 la n, aflați în diferite localități, au o convenție de convocare. Ei vor fi convocați telefonic într-o localitate fixă numită bază. Convenția de transmitere a convocării cuprinde următoarele reguli:

- cu excepția unora, fiecare spion știe ce persoane trebuie să-l contacteze telefonic; pentru a considera convocarea validă și a-i da curs, trebuie să primească semnalul de convocare de la *toate* persoanele de contact;
- cei câțiva spioni care nu așteaptă nici o convocare știu că trebuie să pornească *toți deodată*, la o oră considerată ora 0, spre bază;
- fiecare persoană, imediat ce a ajuns la bază, primește lista persoanelor pe care trebuie să le sune, transmițându-le convocarea;
- durata unei convorbiri telefonice este neglijabilă;
- abia în momentul în care sunt prezente toate cele n persoane la bază poate să înceapă ședința.

Cerință

Cunoscându-se, pentru fiecare spion, durata deplasării până la bază (exprimată în ore) și lista persoanelor care trebuie să îl contacteze, scrieți un program care stabilește dacă se poate realiza convocarea tuturor persoanelor. Dacă este posibil, să se determine:

a) numărul minim de ore care trec până când poate să înceapă ședința;
b) numărul minim de persoane care trebuie transportate în regim de urgență astfel încât ședința să poată începe cu cel puțin o oră mai devreme; numim *transport în regim de urgență* un transport în care, în locul timpului dat, transportul durează cu cel puțin o oră mai puțin;
c) numerele de ordine ale persoanelor care vor fi transportate în regim de urgență.

Date de intrare

Fișierul de intrare **SPIONI.IN**, conține:

- pe prima linie, numărul n de spioni;
- pe a doua linie, n numere naturale $t_1, t_2, \ldots, t_n$, despărțite prin câte un spațiu, numere reprezentând durata deplasării fiecărui spion până la bază (în regim normal), durată exprimată în ore;
- pe fiecare dintre următoarele n linii, câte o succesiune de cel puțin 0 și cel mult n numere naturale despărțite prin câte un spațiu, numerele de pe linia i+2 a fișierului reprezentând lista persoanelor care trebuie să convoace persoana având numărul i, pentru ca aceasta să pornească spre bază.

Datele de ieșire

Pe prima linie a fișierului **SPIONI.OUT** se află numărul -1 dacă nu se poate realiza convocarea, sau numărul de ore după care poate să înceapă ședința, în caz contrar. În cazul în care convocarea este posibilă, fișierul mai conține:

- pe a doua linie, un număr p, reprezentând numărul de persoane care, transportate în regim de urgență, pot devansa ora de începere a ședinței cu cel puțin o oră;
- pe a treia linie, numerele de ordine ale persoanelor care trebuie să fie transportate în regim de urgență, pentru a putea începe ședința mai devreme, numere despărțite prin câte un spațiu.

Restricții

- $1 \le n \le 100$
- $2 \le t_i \le 500$

Exemplu

SPIONI.IN	**SPIONI.OUT** poate conține:
5	18
9 8 3 9 1	1
4	
1 3 5	
2 3	

Timp maxim de execuție
3 secunde/test

P050018: Tezaur

prof. Doru Anastasiu Popescu, Liceul "Radu Greceanu", Slatina

Într-un document arheologic recent descoperit se face referire la un mare tezaur. Datorită faptului că documentul poate fi interpretat în mai multe moduri se face apel la n arheologi care vor studia independent documentul.

La terminarea studiului, fiecare arheolog întocmește o hartă pe care marchează o zonă poligonală închisă și convexă despre care presupune că este locul unde se află tezaurul.

Deoarece fondurile alocate pentru descoperirea tezaurului sunt reduse, se ia hotărârea să se înceapă cercetările pe teren, doar în zona, precizată de toți arheologii.

Cerință

Cunoscând valoarea n și coordonatele vârfurilor zonelor determinate de arheologi, să se determine aria suprafeței de pe hartă, precizată de toți arheologii (intersecția celor n zone).

Date de intrare

Fișierul de intrare **TEZAUR.IN** conține:

n - numărul de zone;
m[1] - numărul de vârfuri pentru zona primului arheolog
$x_{1,1}$ $y_{1,1}$ ... $x_{1,m[1]}$ $y_{1,m[1]}$ - coordonatele vârfurilor zonei primului arheolog (date în sens invers acelor de ceasornic);

...

m[n] - numărul de vârfuri pentru zona ultimului arheolog
x_{n1} y_{n1} ... $x_{nm[n]}$ $y_{nm[n]}$ - coordonatele vârfurilor zonei celui de-al n-lea arheolog (date în sens invers acelor de ceasornic).

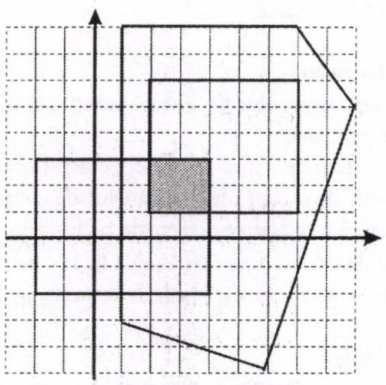
Date de ieşire

Pe prima linie a fişierului **TEZAUR.OUT** se va scrie aria zonei de pe hartă, precizată de toţi arheologii, cu două zecimale.

Dacă nu există suprafaţă comună zonelor celor n arheologi, în fişierul de ieşire se va scrie numărul 0.

Exemplu

TEZAUR.IN	TEZAUR.OUT
3	400.00
4	
-20 30 -20 -20 40 -20 40 30	
5	
10 80 10 -30 60 -50 100 60 70 80	
4	
20 10 70 10 70 60 20 60	



Restricţii

- $1 \le n \le 30$;
- $3 \le m[i] \le 20$, $i \in \{1,2,\ldots,n\}$;
- Coordonatele vârfurilor zonelor sunt numere întregi din intervalul [-1000,1000];
- Numărul din fişierul de ieşire se va scrie cu 2 zecimale.

Timp maxim de execuţie

1 secundă/test

Baraj

P050019: Antene

prof. Dana Lica, Liceul "Ion Luca Caragiale", Ploieşti

Firmele de telefonie mobilă *Dialog* şi *Connex* urmează să-şi instaleze antene satelit într-un spaţiu comun. În acest scop, se realizează o hartă sub forma unui caroiaj dreptunghiular, unde se dau N puncte prin coordonate naturale, reprezentând punctele în care vor fi instalate antene pentru cele două firme.

Managerul *Oficiului Concurenţei* doreşte ca nici una din firme să nu fie dezavantajată. Astfel, ea stabileşte următoarea condiţie care trebuie să fie respectată:

- orice dreaptă paralelă cu axa *Ox* sau *Oy* trasată printr-un punct de coordonate naturale intersectează un număr de antene ale firmei *Connex*, număr care diferă cu cel mult 1 de numărul de antene ale firmei *Dialog* intersectate de aceeaşi dreaptă. Altfel spus, modulul diferenţei dintre numărul antenelor *Connex* şi numărul antenelor *Dialog* situate pe aceeaşi dreaptă este cel mult 1.

Cerinţă

Cunoscându-se coordonatele celor N puncte, să se realizeze o distribuire a tuturor punctelor între cele două firme, astfel încât restricţia impusă de managerul *Oficiului Concurenţei* să fie respectată.

Date de intrare:

- Prima linie a fişierului de intrare **ANTENE.IN** conţine numărul de puncte (N).
- Pe fiecare din următoarele N linii sunt scrise câte două numere x y, despărţite printr-un spaţiu, reprezentând coordonatele fiecărui punct în care se va instala o antenă.
- Coordonatele punctelor sunt numere naturale cuprinse între 0 şi 30000.

Date de ieşire

Fişierul de ieşire **ANTENE.OUT** va conţine N linii:

- pe fiecare linie se vor scrie trei numere despărţite prin câte un spaţiu: x y c; primele două numere (x, y) reprezintă coordonatele unui punct din caroiajul dat, în care este permisă amplasarea unei antene, iar ultimul număr (c) va avea valoarea 1 dacă în acel punct s-a amplasat antena firmei *Dialog*, respectiv -1 dacă s-a amplasat antena firmei *Connex*.
- Ordinea de scriere a punctelor în fişierul de ieşire poate fi oarecare.

Restricţie

$N \le 8000$

Exemplu

ANTENE.IN	ANTENE.OUT
6	1 3 1
4 1	2 3 -1
3 2	2 4 1
1 3	3 4 -1
2 3	4 1 1
2 4	3 2 1
3 4	

Timp maxim de execuţie

2 secunde/test

P050020: Solitaire

prof. Emanuela Cerchez, Liceul de Informatică, Iași

Un jucător solitar a inventat următorul joc. El desenează pe o foaie de hârtie n puncte.

Jocul constă în construirea unor segmente după următoarele reguli:
- orice segment unește două din cele n puncte;
- orice punct poate să fie extremitatea a cel mult un segment și nu poate să aparțină interiorului segmentelor;
- oricare două segmente nu se intersectează;
- jocul continuă până când nu se mai poate trasa nici un astfel de segment.

Scopul jocului este de a trasa segmente astfel încât, în final, numărul de puncte rămase izolate (care nu reprezintă extremitățile unor segmente) să fie minim.

Cerință

Scrieți un program care să joace acest joc.

Date de intrare

Din fișierul de intrare **JOC.IN** se citesc:
n - numărul de puncte;
x_1 y_1 - coordonatele carteziene ale primului punct;
x_2 y_2 - coordonatele carteziene ale celui de-al doilea punct;
...
x_n y_n - coordonatele carteziene ale punctului n.

Date de ieșire

Rezultatele se vor scrie în fișierul **JOC.OUT** sub forma:
P - numărul de puncte rămase izolate;
K - numărul de segmente construite;
x_{11} y_{11} x_{12} y_{12} - coordonatele carteziene ale extremităților primului segment;
x_{21} y_{21} x_{22} y_{22} - coordonatele extremităților celui de-al doilea segment;
...
x_{k1} y_{k1} x_{k2} y_{k2} - coordonatele carteziene ale extremităților segmentului k.

Restricție

$n \leq 200$

Timp maxim de execuție

1 secundă/test

P050021: Moșteniri duble

prof. Roxana Tâmplaru, Liceul "Ștefan Odobleja", Craiova

Ministrul de Finanțe din Simot a decis să pună impozit pe moștenirile duble.

În registrul de stare civilă (cu ajutorul căruia se va stabili câte impozite se pot încasa) apar n persoane, numerotate de la 1 la n.

O persoană care nu figurează ca moștenitor al nici unei persoane este denumită *strămoș*.

O persoană care este moștenitor direct al cuiva, dar nu are nici un moștenitor (direct sau indirect) este denumită *terminal*.

Celelalte persoane care moștenesc de la una sau mai multe persoane și au unul sau mai mulți moștenitori sunt denumiți *tranzienți*.

Numim *succesiune de moștenitori* o secvență de persoane $p_1, p_2, \ldots, p_k$, unde p_1 este strămoș, $p_2, \ldots, p_{k-1}$ sunt tranzienți, p_k este terminal, iar p_i este moștenitorul direct al lui p_{i-1}, pentru oricare $i \in \{2, \ldots, k\}$.

Se impozitează doar persoanele terminale, beneficiare de moșteniri duble, știind că o persoană plătește câte un impozit pentru fiecare moștenire dublă. Numim *moștenire dublă* a unui terminal două succesiuni distincte de moștenitori care au același strămoș. Se consideră că două succesiuni sunt distincte dacă există cel puțin o persoană tranzientă care apare într-o succesiune și nu apare în cealaltă.

Dacă persoana i este moștenitorul direct sau indirect al persoanei j, nu se poate întâmpla ca persoana j să fie moștenitor direct sau indirect al persoanei i.

Cerință

Scrieți un program care stabilește numărul total de impozite care pot fi încasate.

Date de intrare

- Prima linie a fișierului de intrare **MOST.IN** conține numărul de persoane n.
- De pe următoarele n linii se citesc, pentru fiecare persoană numerele de ordine ale persoanelor de la care persoana respectivă a moștenit direct (de pe linia i+1 se citesc numerele de ordine ale persoanelor de la care a moștenit direct persoana i). Fiecare linie se termină cu zero.

Date de ieșire

În fișierul de ieșire **MOST.OUT** se va scrie numărul total de impozite.

Restricție

$N \leq 200$

Exemplul 1

MOST.IN	MOST.OUT
14	5
0	
0	
0	
0	
1 0	
5 0	
8 0	
1 2 0	
1 3 0	
3 4 0	
6 7 12 0	
9 0	
10 0	
7 12 13 0	

Explicație

Sunt 5 impozite pe 5 moșteniri duble.

Primul impozit îl plătește persoana 11 care a moștenit de la persoanele 6 și 7 (care au persoana 1 ca strămoș). Al doilea impozit îl plătește tot persoana 11 datorită moștenirilor de la persoanele 6 și 12 (care au persoana 1 ca strămoș). Al treilea impozit îl va plăti tot persoana 11 datorită moștenirilor primite de la persoanele 7 și 12 (care au pe persoana 1 ca strămoș).

Al patrulea impozit îl plătește persoana 14 care a moștenit de la persoanele 7 și 12 (care au persoana 1 ca strămoș). Al cincelea impozit, îl plătește tot persoana 14 care moștenește de la persoanele 12 și 13 (care au persoana 3 ca strămoș).

Exemplul 2

MOST.IN	MOST.OUT
14	0
0	
0	
0	
0	
1 0	
5 0	
8 0	
2 0	
3 0	
4 0	
6 7 0	
9 0	
10 0	
12 13 0	

Exemplul 3

MOST.IN	MOST.OUT
4	1
0	
1 0	
1 2 0	
3 0	

Explicație

Impozitul este plătit de persoana 4 pentru care cele două succesiuni distincte sunt: 1, 2, 3, 4 și 1, 3, 4, strămoșul comun fiind persoana 1.

Timp maxim de execuție

2 secunde/test

P050022: Multiplu

prof. Ovidiu Domșa, Colegiul "Horea, Cloșca și Crișan", Alba Iulia

Se dă un număr natural n și k cifre distincte nenule.

Cerință

Să se găsească cel mai mic număr natural m, multiplu al lui n, format numai cu cifrele date.

Date de intrare

- Prima linie a fișierului de intrare **MULTIPLU.IN** conține valorile n și k separate printr-un spațiu.
- A doua linie conține cele k cifre separate prin câte un spațiu.

Date de ieșire

Dacă problema are soluție, în fișierul **MULTIPLU.OUT** se va scrie numărul m. În cazul în care nu există soluție, în fișierul **MULTIPLU.OUT** se va scrie mesajul NU.

Restricție

$1 \le n \le 10000$, $1 \le k \le 9$

Exemplul 1

MULTIPLU.IN	MULTIPLU.OUT
3458 4	31122
1 3 5 2	

Exemplul 2

MULTIPLU.IN	MULTIPLU.OUT
15 5	NU
1 3 4 7 9	

Timp maxim de execuție

1 secundă/test

P050023: Parantezări

Mihai Oltean, doctorand Universitatea Babeș Bolyai, Cluj

Se consideră mulțimea șirurilor de N perechi de paranteze rotunde împerecheate corect.

Această mulțime se consideră ordonată lexicografic crescător, știind că '(' < ')'.

Cerință

Dându-se un șir de paranteze se cere să se determine numărul lui de ordine, știind că prima configurație

((((...(())...))))

are numărul de ordine 1.

Date de intrare

Prima linie a fișierului de intrare **PAR.IN** conține numărul N, iar a doua linie conține șirul de paranteze.

Date de ieșire

În fișierul de ieșire **PAR.OUT** se va scrie un număr întreg. Acest număr trebuie să reprezinte numărul de ordine al parantezării date, în șirul ordonat lexicografic crescător al tuturor parantezărilor corecte formate din N perechi de paranteze.

Restricție

$1 \le n \le 30$

Exemplul 1

PAR.IN

```
3
((()))
```

PAR.OUT

```
1
```

Exemplul 2

PAR.IN

```
4
() () () ()
```

PAR.OUT

```
14
```

Exemplul 3

PAR.IN

```
2
() ()
```

PAR.OUT

```
2
```

Exemplul 4

PAR.IN

```
4
() () (())
```

PAR.OUT

```
13
```

Timp maxim de execuție
1 secundă/test

P050024: Vânătorii

Cristian Cadar, student Universitatea Politehnică București
Marius Vlad, student Universitatea Politehnică București

Cristi și Marius s-au hotărât într-o zi să se ducă la vânătoare. Cristi are o pușcă destul de prăpădită cu care poate vâna doar lupi. Marius în schimb, care tocmai și-a luat bursa, are o pușcă performantă cu care poate vâna și lupi și mistreți. Ajunși în pădure, ei își dau seama că vânătoarea nu e treabă ușoară; ei trebuie să alerge mult după fiecare animal. Având la dispoziție un număr de doar T minute alocate pentru vânătoare și dorind să vâneze în acest timp cât mai multe animale (lupi și mistreți) ei au calculat, pe baza unor informații date de pădurar, câte minute trebuie să alerge după fiecare animal în parte.

Cerință
Scrieți un program care determină câte animale trebuie să vâneze Cristi și Marius pentru ca împreună să aducă acasă un număr maxim de animale.

Date de intrare
- Prima linie a fișierului de intrare **VANATORI.IN** conține numărul T, reprezentând durata maximă a vânătorii.
- Pe cea de-a doua linie sunt scrise două numere L și M, reprezentând numărul de lupi respectiv de mistreți, despărțite printr-un spațiu.
- Pe cea de-a treia linie se găsesc L numere întregi, fiecare număr t_i (i=1,L) reprezentând timpul în care poate fi vânat fiecare dintre cei L lupi.
- Cea de-a patra linie conține M numere întregi, fiecare număr u_i (i=1,M) reprezentând timpul în care poate fi vânat fiecare dintre cei M mistreți.

Date de ieșire
Pe prima linie a fișierului de ieșire **VANATORI.OUT** se va scrie un singur număr, reprezentând numărul maxim de animale pe care le pot vâna Marius și Cristi.

Restricții:
- $1 \leq T \leq 300$
- $0 \leq L \leq 600$
- $0 \leq M \leq 600$
- $0 \leq t_i, u_i \leq T$

Exemplu
VANATORI.IN

```
5
4 2
2 1 1 2
5 4
```

VANATORI.OUT

```
5
```

Explicație
Cristi vânează primul, al doilea și al patrulea lup, iar Marius al treilea lup și al doilea mistreț.

Timp maxim de execuție
10 secunde/test

Din păcate, la închiderea ediției, redacția GInfo nu a avut la dispoziție enunțurile problemelor în forma lor finală, așa cum acestea au fost prezentate concurenților. Sperăm ca eventualele diferențe să nu creeze dificultăți în înțelegerea lor.

CONCURS

Prima fază a concursului organizat de Gazeta de Informatică a luat sfârșit. Se pare că ultima rundă a fost cea mai dificilă, dovadă numărul mic de concurenți care au avut curajul să încerce să rezolve cele trei probleme. În urma acestei runde a fost alcătuit și clasamentul final, primii zece clasați participând la faza a doua a competiției.

Faza prin e-mail a concursului ***Bursele Agora*** s-a încheiat. Primii zece clasați la sfârșitul acestei faze vor participa pe data de 17 iunie 2000 la etapa finală care se va desfășura la Liceul de Informatică "Tiberiu Popoviciu" din Cluj-Napoca. După șase runde prima parte a clasamentului arată astfel:

1. Adrian Cârcu, Bistrița	**1528**
2. Mihai Pătrașcu, Craiova	**1480**
3. Ioana Ileană, Alba Iulia	**1353**
4. Cristian Cadar, București	**1295**
5. Mihai Stroe, București	**1281**
6. Daniel Dumitran, București	**1229**
7. Vlad Vâlceanu, Arad	**1190**
8. Ciprian Baicu, Dr. Tn. Severin	**1126**
9. Csaba Patcas, Oradea	**1108**
10. David Andrei, Rm. Vâlcea	**1048**

Nu putem decât să-i felicităm pe primii clasați și să le urăm succes la etapa finală.

Mulțumim tuturor celor care și-au încercat puterile la concursul nostru și sperăm că se vor număra printre participanții la ediția viitoare. Ne place să credem că ***Bursele Agora*** a fost un concurs util pentru atât pentru elevii care doresc să se pregătească pentru olimpiadele de informatică cât și pentru studenții care vor participa la concursurile studențești de programare, în special la ACM.

Nu putem să nu amintim rezultatele obținute de cei care se situează pe locurile imediat următoare:

11. Gabriel Pârvan, Râmnicu Sărat	*1012*
12. Cosmin Silvestru Negruseri, Bistrița	*980*
13. Gabriel Erzse, Oradea	*978*
14. Marian Anghel, Buftea	*961*
15. Victor Asavei, Drobeta Turnu-Severin	*955*
16. Virgil Șerbănuță, București	*844*
17. Cristian Alexandrescu, Botoșani	*844*
18. Mihai Dondera, București	*811*
19. Mihai Maerean, Slobozia	*773*
20. Ciprian Cană, Vaslui	*756*

Îi felicităm și pe ei și le urăm mai mult noroc data viitoare.

Acum, la final, merită amintiți și cei care au încercat să rezolve fiecare problemă propusă, chiar dacă nu au reușit întotdeauna. Cei care au trimis rezolvări la toate cele 18 probleme propuse sunt ***Adrian Cârcu***, ***Mihai Pătrașcu***, ***Vlad Vâlceanu***, ***Csaba Patcas***, ***Andrei David***, ***Anghel Marian***, ***Mihai Maerean*** și ***Ciprian Cană***. Probabil că dacă nu ar fi existat problema ***Rubik*** de la runda a șasea numărul acestora ar fi fost mult mai mare.

Să revenim acum la runda a șasea. Problemele au fost într-adevăr foarte dificile, una dintre ele (Rubik) fiind de mult cunoscută ca fiind o problemă la care nu s-a găsit încă soluție. Primii clasați în clasamentul rundei a șasea sunt următorii:

1. Adrian Cârcu, Bistrița	*200*
1. Mihai Stroe, București	*200*
3. Ioana Ileană, Alba Iulia	*195*
4. Mihai Pătrașcu, Craiova	*174*
5. Cosmin Silvestru Negruseri, Bistrița	*173*
6. Vlad Vâlceanu, Arad	*139*
7. Victor Asavei, Dr. Tn. Severin	*135*
7. Liviu Lalescu, Craiova	*135*
9. Daniel Dumitran, București	*116*
10. Csaba Patcas, Oradea	*105*
10. Mihai Maerean, Slobozia	*105*

Așa cum am mai spus, primii zece clasați după cele șase runde vor participa la faza finală din 17 iunie. Pentru ceilalți ediția de anul acesta a concursului Bursele Agora a luat sfârșit. Îi așteptăm la ediția următoare când, sperăm, vom avea participanți și de peste hotare.

Regulamentul de desfășurare (în limba engleză) a celei de-a doua ediții îl veți putea găsi în ediția specială *GInfo* care va apărea cu ocazia *Olimpiadei de Informatică a Europei Centrale* care se va desfășura între 24 și 31 august la Cluj-Napoca. Versiunea în limba română a regulamentului va apărea în numărul 10/6 (octombrie 2000) al *GInfo*.

Urăm de pe acum succes tuturor viitorilor participanți!!!.

Runda 6, problema 1

Galbeni

Această problemă a fost propusă, în diverse variante, la diferite concursuri de informatică din România, deci ar fi trebuit să fie cunoscută participanților. Totuși problema era destul de dificilă dacă nu se cunoștea algoritmul, și punctajul mediu a fost mai mic decât ne-am așteptat: 57.85 puncte. Ne-a surprins și faptul că au existat doar cinci concurenți care au obținut 100 de puncte.

Acum două numere îl criticam pe Mihai Pătrașcu din Craiova pentru felul în care își scria sursele. Așa cum ne-am așteptat, replica nu s-a lăsat întârziată și concurentul ne-a trimis o rezolvare demnă de toată lauda pentru această problemă, inclusiv o descriere excepțională a algoritmului folosit.

Presupunem că avem o soluție a problemei (configurația A_0, B_0, C_0 am transformat-o in A_k, B_k, C_k cu $A_k=B_k$). Mai aplicăm o operație, folosind cele două grămezi egale (A_k, $B_k=A_k$, C_k) și obținem $2\cdot A_k$, 0, C_k). Deci problema se poate reformula, cerându-se să se obțină 0 într-o grămadă (evident, dacă am obținut 0 într-o grămadă, înainte de ultima operație de scădere aveam două grămezi identice).

Vom căuta un șir de operații (A_i, B_i, $C_i \rightarrow A_k$, B_k, C_k cu $k > i$) astfel încât $min(A_i, B_i, C_i) > min(A_k, B_k, C_k)$. Cum A_i, B_i, C_i sunt numere naturale și nu există șiruri descrescătoare infinite de numere naturale, minimul sumelor din cele trei grămezi va ajunge într-un număr finit de pași la 0.

Fie configurația A_i, B_i, C_i. Presupunem că $A_i \le B_i \le C_i$ (deci, $min(A_i, B_i, C_i) = A_i$). Vom transforma configurația într-o configurație A_k, B_k, C_k cu $B_k = rest[B_i/A_i]$. Evident $B_k<A_i$, deci $B_k<min(A_i, B_i, C_i)$ și $min(A_k, B_k, C_k)<min(Ai, B_i, C_i)$. Pentru această transformare, vom calcula $x=[B_i/A_i]$. Analizăm biții reprezentării în baza 2 a lui x pornind de la cel mai puțin semnificativ. Orice bit cu valoarea 1 îl anulăm printr-o transformare de forma A_i, B_i, $C_i \rightarrow 2\cdot A_i$, B_i-A_i, C_i). Pentru un bit de 0 trebuie să nu modificăm B_i, dar să-l deplasăm la stânga biții lui A_i cu o poziție. Aceasta se realizează simplu printr-o transformare de forma A_i, B_i, $C_i \rightarrow A_i\cdot 2$, B_i, C_i-A_i.

Este evident că, după ce am parcurs toți biții lui x, în grămada B vom avea $rest[B_i/A_i]$ galbeni. Trebuie să ne asigurăm că aceste transformări nu generează numere negative de galbeni. Este evident că $A_k>0$ (A se dublează la fiecare operație) și $B_k\ge 0$ (B scade la fiecare operație, dar valoarea minimă este $rest[B_i/A_i]\ge 0$). Șirul C_k este de asemenea descrescător, deci va trebui să calculăm valoarea sa minimă. Observăm că $A_i+B_i+C_i=A_k+B_k+C_k$. Dar $B_k=rest[B_i/A_i]$. Deci $A_i+(B_k+A_i\cdot x)+C_i=A_k+B_k+C_k$ și, scăzând

CR6P1 Top 10

1. Adrian Cârcu, Bistrița	100
1. Cosmin Silvestru Negruseri, Bistrița	100
1. Gabriel Pârvan, Râmnicu Sărat	100
1. Mihai Pătrașcu, Craiova	100
1. Mihai Stroe, București	100
6. Mihai Dondera, București	95
6. Ioana Ileană, Alba Iulia	95
8. Daniel Dumitran, București	70
9. Victor Asavei, Drobeta Turnu-Severin	60
10. David Andrei, Râmnicu Vâlcea	55

B_k din ambii membri, avem: $A_i\cdot(1+x)+C_i=A_k+C_k$. Pentru fiecare bit al lui x, A_i își dublează valoarea, deci $A_k\le A_i\cdot x\cdot 2$. Dar, $C_k=C_i+A_i\cdot(1+x)-A_k$, deci $C_k\ge C_i+A_i\cdot(1+x)-A_i\cdot x\cdot 2$, adică $C\ge C_i-A_i\cdot(x-1)$. Dar, $B_i\ge A_i\cdot x$ și $C\ge B_i$, deci $C_i\ge A_i\cdot x$. Rezultă că $C_k\ge A_i$, deci $C_k\ge 0$. În concluzie aceste transformări sunt valide și problema poate fi rezolvată, folosind acest algoritm.

O estimare a complexității algoritmului indică ordinul $O(log\ N\cdot k)$, unde $N=max(A_0, B_0, C_0)$ și k este valoarea minimă pentru care $F(k)\ge N$ (F este șirul lui *Fibonacci*). Folosind un rezultat datorat lui *A. de Moivre*, ajungem la $k=O(log\ N)$. Deci algoritmul are complexitatea $O(log^2\ N)$.

Listing GALBENI.CPP

```
#pragma option -3 -r -Z -O2 -a
#include <stdio.h>

int main(){
  unsigned long na,nb,nc,*a,*b,*c,x;
  int ia,ib,ic;
  fscanf(fopen("GOLD.IN","r"),"%lu%lu%lu",
                              &na,&nb,&nc);
  for (freopen("GOLD.OUT","w",stdout);;){
    if (na>nb) a=&nb,ia=2,b=&na,ib=1;
    else a=&na,ia=1,b=&nb,ib=2;
    if (nc>=*b) c=&nc,ic=3;
    else{ c=b,ic=ib;
          if (nc>=*a) b=&nc,ib=3;
          else b=a,ib=ia,a=&nc,ia=3; }
    for (x=*b/*a;x;x>>=1){
      if (x&1){ if (*a==*b) return(0);
                *b-=*a,*a<<=1;
                printf("%d %d\n",ib,ia); }
      else{ if (*a==*c) return(0);
            *c-=*a,*a<<=1;
            printf("%d %d\n",ic,ia);
          }
    }
  }
}
```

Rubik

Aceata este problema "fără soluție" de care am vorbit. Au fost doar 12 participanți care au încercat să o rezolve, după cum puteți vedea în lista alăturată. Punctajul maxim a fost de 30 de puncte, iar cel mediu de doar 9.58 puncte.

Dacă s-ar cunoaște valoarea la care trebuie să se ajungă, problema s-ar reduce la rezolvarea unui sistem de $m \times n$ ecuații cu $m \times n$ necunoscute. Fiecărei ecuații i-ar corespunde un element al grilei, valoarea finală a fiecărui element fiind obținută adunând la valoarea sa inițială numărul obținut prin însumarea numerelor k, corespunzătoare operațiilor de pe poziția sa și a vecinilor săi pe verticală sau orizontală. Deci, în principiu, o ecuație ar avea forma $b_{ij}=a_{ij}+k_{ij}+k_{i-1,j}+k_{i+1,j}+k_{i,j-1}+k_{i,j+1}$, unde b_{ij} este valoarea finală care trebuie obținută pentru toate elementele grilei, a_{ij} este valoarea inițială a elementului, iar k_{xy} valoarea adunată pentru operația efectuată asupra poziției (x, y). Singurele ecuații care nu au această formă sunt cele corespunzătoare pozițiilor de pe marginea grilei, caz în care vom avea două elemente în ecuație (dacă avem o grilă cu o singură linie sau o singură coloană, pentru elementele de la extreme), trei elemente (pentru restul elementelor dintr-o grilă cu o singură linie sau o singură coloană și pentru elementele din colțurile unei grile "normale") sau patru elemente (pentru elementele de pe marginile unei grile normale).

În mod normal, ar trebui să rezolvăm un sistem de maxim n=10000 de ecuații, fiecare având maxim k=5 necunoscute și există algoritmi numerici care rezolvă astfel de sisteme (după transformarea sistemului în unul echivalent care respectă anumite condiții) într-un timp de ordinul $O(m \times k)$.

Datorită faptului că sistemul rezultat este unul particular, el poate fi transformat într-unul echivalent care să aibă maxim 100 de ecuații, fiecare având maxim 100 de necunoscute, sistem care poate fi rezolvat într-un timp acceptabil, folosind o metodă standard cum ar fi cea a lui *Gauss*.

Tot ceea ce am spus anterior este valabil numai dacă am cunoaște valoarea care trebuie să fie obținută pentru elementele grilei, ceea ce nu este cazul nostru. Totuși știm că această valoare este un număr natural cuprins între 1 și 1000, deci am putea să încercăm să rezolvăm sistemul pentru toate aceste valori posibile. Dar dacă vom verifica pentru toate cele 1000 de valori, timpul de execuție ar crește de 1000 de ori, iar programul nu se va mai executa într-un timp acceptabil. O posibilă soluție ar fi să încercăm să "ghicim" care ar fi valoarea la care trebuie să se ajungă. Datorită faptului că printr-o operație elementele grilei pot să crească sau să descrească, pe cazul mediu valoarea care trebuie obținută ar trebui să fie apropiată de media elementelor grilei inițiale. Evident, acest lucru nu este valabil pentru toate cazurile deoarece, dacă trebuie efectuate operații pentru care valorile k au toate același semn și sunt relativ mari în valoare absolută, valoarea care trebuie obținută nu este apropiată de media elementelor. Totuși, datorită faptului că datele de test folosite au fost generate aleator, o astfel de abordare a problemei ar fi dus la obținerea de punctaje mai mari.

Datorită faptului că problema nu are încă o soluție care să de-a rezultate pe toate testele posibile nu vă putem prezenta nici un algoritm care să o rezolve pentru toate cazurile. Poate că data viitoare vom găsi un concurent care să o rezolve; până atunci problema Rubik va rămâne o provocare pentru toți informaticienii. Nu putem decât să sperăm că cineva va reuși să o rezolve până la urmă.

Așteptăm cu interes o rezolvare corectă și vă promitem că o vom publica imediat ce ea va apărea!!!

Au încercat...

Liviu Lalescu, Craiova	30
Vlad Vâlceanu, Arad	20
Ciprian Baicu, Dr. Tn. Severin	15
Adrian Cârcu, Bistrița	15
David Andrei, Râmnicu Vâlcea	10
Marian Anghel, Buftea	5
Victor Asavei, Dr. Tn. Severin	5
Mihai Maerean, Slobozia	5
Csaba Patcas, Oradea	5
Mihai Pătrașcu, Craiova	5
Ciprian Cană, Vaslui	0
Mihai Dondera, București	0

Runda 6, problema 3

Expresie

Aceasta este o problemă pentru care nu se cunoaște încă o rezolvare polinomială. Totuși algoritmii găsiți de concurenți, marea lor majoritate probabiliști, s-au comportat destul de bine, punctajul mediu fiind de 58.56 puncte.

Chiar și soluția oficială folosea un algoritm genetic care însă funcționa foarte bine. Am avut totuși doi concurenți care au obținut punctaj maxim: *Ioana Ileană* din Alba Iulia și *Mihai Stroe* din București.

Punctajul "real" al Ioanei Ileana a fost de 99.95 puncte, la un singur test expresia obținută de ea având valoarea de 1234567889 în loc de 1234567890.

Punctajul "real" al lui Mihai Stroe a fost de 99.90 puncte, la același test valoarea expresiei furnizate de programul său fiind 1234567888.

Datorită celor 0.05 puncte în plus vă prezentăm soluția propusă de Ioana Ileană.

CR6P3 Top 10

1.	Ioana Ileană, Alba Iulia	100
1.	Mihai Stroe, București	100
3.	Adrian Cârcu, Bistrița	85
4.	Vlad Vâlceanu, Arad	79
5.	Silvestru Negruseri, Bistrița	73
6.	Liviu Lalescu, Craiova	70
6.	Victor Asavei, Dr. Tn. Severin	70
6.	Csaba Patcas, Oradea	70
9.	Mihai Pătrașcu, Craiova	69
10.	Mihai Maerean, Slobozia	55

Listing EXPRESIE.PAS

```
program Expresie;
const nmax=20;
      opmax=105;
      tmax=130;
      maxi=MaxLongint;
type sir=array[0..20] of Byte;

var f:Text;
    n,i,j,k,x,y:Integer;
    a:array[0..nmax] of Longint;
    last1,last2,nrop,pozinit,where:
                    array[0..nmax] of Byte;
    aux,best,s,alarm,solutie:Longint;
    found:Boolean;
    auxsgn:Char;
    nrmax,nrtotal:Byte;
    solrez,solauxrez:
                array[0..opmax] of Longint;
    solop,solauxop:array[0..opmax] of sir;
    time:Longint absolute $0000:$046C;

procedure citire;
begin
  Assign(f,'INPUT.TXT');
  Reset(f);
  Readln(f,s);
  while not Seekeoln(f) do
  begin
    n:=n+1;
    Read(f,a[n]);
    pozinit[n]:=n
  end;
  Close(f)
end;

procedure scrie(i:Byte);
begin
  if nrop[i]=0 then
  begin
    Write(f,a[i]);
    Exit
  end;
  scrie(where[last1[i]]);
  Write(f,'+');
  scrie(where[last2[i]])
end;

procedure afis(i:Byte);
var j,t,k:Byte;
```

```
begin
  t:=0;
  for j:=1 to n do
    for k:=1 to solop[i][j] do
    begin
      t:=t+1;
      if t>1
        then Write(f,'*');
      if nrop[j]<>0
        then Write(f,'(');
      scrie(j);
      if nrop[j]<>0
        then Write(f,')')
    end
end;

procedure afisare;
var i:Byte;
begin
  if nrmax>100 then Exit;
  if Abs(s-solutie)>=best then Exit;
  best:=Abs(s-solutie);
  Assign(f,'OUTPUT.TXT');
  Rewrite(f);
  afis(1)
  for i:=2 to solrez[0] do
  begin
    Write(f,'+');
    afis(i)
  end;
  Close(f);
  if best=0 then Halt
end;

procedure avanseaza;
var x,y:sir;
    valx,valy:Longint;
    i,j,nop,nopy:Integer;
begin
  valx:=1;
  for i:=0 to nmax do x[i]:=0;
  nop:=-1;
  i:=n;
  while (i>0) and (aux div valx<a[i]) do
    i:=i-1;
  while (i<>0) do
  begin
    j:=Random(i)+1;
    if nop=-1 then nop:=0;
    valx:=valx*a[j];
    nop:=nop+1+nrop[j];
    x[j]:=x[j]+1;
    while (i>0) and (aux div valx<a[i]) do
      i:=i-1;
    while (i>0) and (a[i]=1) do i:=i-1
  end;
  if i=0 then
    if nop<>-1 then
      if nrmax+nop+1<=100 then
      begin
        solrez[0]:=solrez[0]+1;
        solop[solrez[0]]:=x;
        nrmax:=nrmax+nop+1;
        solutie:=solutie+valx;
        aux:=aux-valx;
        y:=x;
        nopy:=nop;
        valy:=valx;
        afisare;
        if maxi div valx<a[1]
          then Exit;
        j:=1;
        if nop=-1 then nop:=0;
        valx:=valx*a[j];
        nop:=nop+1+nrop[j];
        x[j]:=x[j]+1;
        while (i>0)and(aux div valx<a[i])do
          i:=i-1;
        solrez[0]:=solrez[0]-1;
        nrmax:=nrmax-nopy-1;
        x[j]:=x[j]-1;
        solutie:=solutie-valy;
        solrez[0]:=solrez[0]+1;
        solrez[solrez[0]]:=valx;
        solop[solrez[0]]:=x;
        solutie:=solutie+valx;
        nrmax:=nrmax+nop;
        afisare;
        solutie:=solutie-valx;
        nrmaX:=nrmax-nop-1;
        nrmax:=nrmax+nopy+1;
        solutie:=solutie+valy
      end
      else aux:=-1;
  if a[1]+solutie>s
    then aux:=-1
end;

procedure calcul;
begin
  if time>alarm then Halt;
  for i:=0 to opmax do
    solrez[i]:=0;
  solutie:=0;
  aux:=s;
  nrmax:=0;
  while (aux>0) do
  begin
    avanseaza;
    afisare
  end
end;
```

```
procedure prelucrare;
begin
  i:=0;
  while (n<20) and not found do
  begin
    i:=i+1;
    x:=Random(n)+1;
    y:=Random(n)+1;
    aux:=a[x]+a[y];
    k:=0;
    if aux>s then k:=1
    else
      for j:=1 to n do
        if a[j]=aux then
        begin
          k:=1;
          j:=n
        end;
    if k=0 then
    begin
      n:=n+1;
      pozinit[n]:=n;
      nrop[n]:=nrop[x]+nrop[y]+1;
      last1[n]:=x;
      last2[n]:=y;
      a[n]:=aux
    end;
    if i>1000
    then found:=true
  end;
```

```
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if a[i]>a[j] then
      begin
        aux:=a[i];
        a[i]:=a[j];
        a[j]:=aux;
        aux:=last1[i];
        last1[i]:=last1[j];
        last1[j]:=aux;
        aux:=last2[i];
        last2[i]:=last2[j];
        last2[j]:=aux;
        aux:=pozinit[i];
        pozinit[i]:=pozinit[j];
        pozinit[j]:=aux;
        aux:=nrop[i];
        nrop[i]:=nrop[j];
        nrop[j]:=aux
      end;
  for i:=1 to n do
    where[pozinit[i]]:=i;
  best:=s;
  while 1=1 do calcul
end;

Begin
  alarm:=time+tmax;
  citire; prelucrare
End.
```

news...

Renunțare

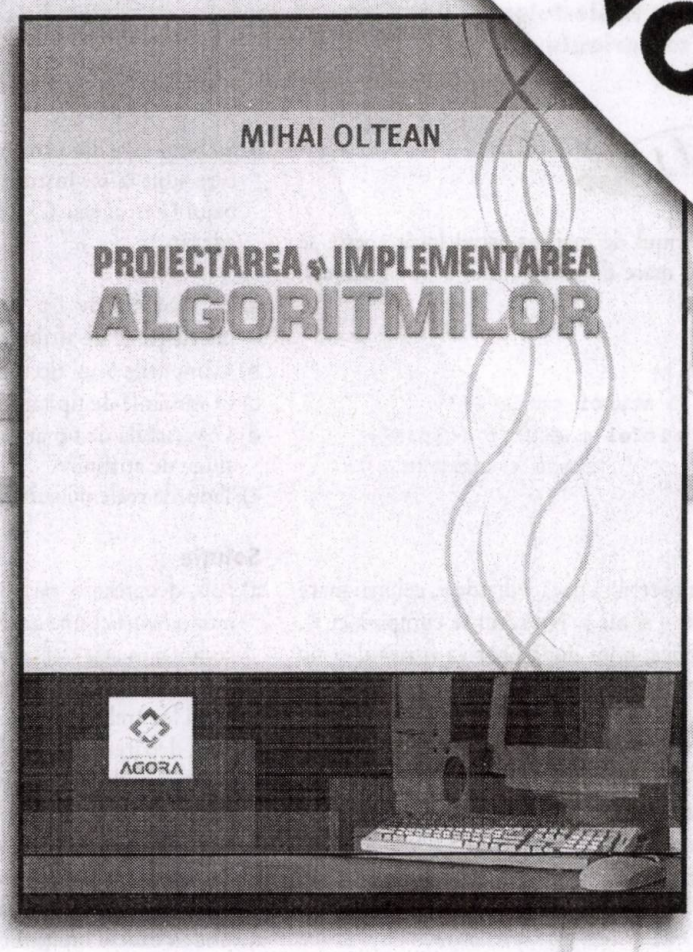
Inprise Corporation a anunțat că fuziunea scontată cu Corel Corporation nu va mai avea loc. Companiile s–au înțeles să renunțe reciproc la cheltuielile prevăzute în caz de terminare a contractului preconizat. Evenimentul se datorează modificării semnificative a situației pe piață a celor două companii.

Specificații pentru Itanium

Compania Intel a publicat pe Internet specificațiile pentru viitoarele procesoare Itanium, încercând astfel să ușureze efortul programatorilor "open source" pentru construirea aplicațiilor destinate noului cip. Aceste specificații, care conțin informații tehnice legate de subsistemele cipului, structura "cache" și alte detalii de microarhitectură, de obicei nu sunt făcute publice înaintea lansării pe piață a cipului. Oricum, Intel și-a schimbat politica și strategia pentru a implementa mai ușor și mai rapid schimbările ce vor fi făcute odată cu lansarea cipului Itanium, fiind silită în mare parte și de popularitatea uriașă de care se bucură sistemul de operare "open source" Linux.

COMPUTER LIBRIS AGORA

NOUTATE



Culegerea constituie un instrumentar pentru cei care doresc să intre în posesia secretelor proiectării algoritmilor. Va fi util când vă pregătiți pentru teză, pentru examenul de bacalaureat sau pentru admitere. Veți găsi în culegere și probleme care pot fi analizate și rezolvate în cercurile de elevi și care vor fi utile în pregătirea pentru concursurile școlare.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

BACALAUREAT 2000

Prezentarea subiectelor de Bacalaureat propuse în cartea Bac Info' 2000 continuă cu varianta nr. 20.

Subiectul 1

1. Modificați algoritmul de mai jos astfel încât acesta să calculeze cel mai mare divizor comun a două numere naturale:

```
funcția cmmdc(m,n)
  dacă [m/n] = 0 atunci cmmdc ← 1
                 altfel m ← n; r ← [m/n];
                        cmmdc ← cmmdc(m,r)
  sfârșit dacă
```

Soluție:

În primul rând, se observă că dacă n divide m, cel mai mare divizor comun va fi n și nu 1. Nu câtul se compară cu 0, ci restul. De asemenea, noul împărțitor va fi restul și nu câtul. În rest, recurența se asigură simplu, ca în funcția de mai jos:

```
funcția cmmdc(m,n)
  dacă rest[m/n]=0
    atunci cmmdc ← n
    altfel cmmdc ← cmmdc(n,rest[m/n])
  sfârșit dacă
```

2. Rescrieți secvența de program de mai jos, utilizând o structură repetitivă cu număr cunoscut de pași.

```
citește n
d ← 2
  cât timp d ≤ n/2 execută
    dacă n mod d = 0 atunci scrie d
    d ← d + 1
  sfârșit cât timp
```

Soluție

```
citește n
  pentru d=2,n/2 execută
    dacă n mod d = 0 atunci scrie d
  sfârșit pentru
```

3. Stabiliți care din următoarele afirmații sunt adevărate și care sunt false. Justificați răspunsurile și precizați limbajul (*Pascal* sau *C*) pentru care ați stabilit valorile de adevăr.

a) O variabilă de tip tablou nu poate să apară într-o instrucțiune de atribuire.
b) Câmpurile unui tip articol trebuie să fie de tip ordinal.
c) O variabilă de tip tablou nu poate fi citită.
d) O variabilă de tip articol poate să apară într-o instrucțiune de atribuire.
e) Tipurile reale nu sunt tipuri ordinale.

Soluție

a) *Fals*, deoarece o variabilă de tip tablou poate să apară într-o instrucțiune de atribuire de forma `t:=a;` (`t=a`) cu condiția ca atât `t` cât și `a` să fie de același tip. Pentru limbajul *Pascal* în cazul în care acest tip este anonim, declararea lor trebuie să se facă într-o singură instrucțiune.
b) *Fals*, deoarece câmpurile unui articol pot avea orice tip.
c) *Adevărat*, deoarece tablourile se citesc element cu element. Altfel spus, dacă `t` este un tablou, nu îl putem citi pe `t`, decât elementele sale `t[i]`.
d) *Adevărat*, justificarea este similară cu cea de la a).
e) *Adevărat*, deoarece numerele reale nu au număr de ordine în cadrul mulțimii numerelor reale.

4. Următoarea secvență de instrucțiuni stabilește dacă un număr natural `n>1` este prim sau nu. Stabiliți posibilitățile de optimizare care se pot aduce algoritmului. Scrieți varianta nouă, justificând modificarea.

```
citește n
d ← 2;
prim ← true
pentru d=1,n/2 execută
  dacă n mod d = 0
    atunci scrie 'Nu e prim'
           prim ← false
  sfârșit dacă
sfârșit pentru
dacă prim atunci scrie 'Prim'
```

Soluție

Prima observație care trebuie fructificată se referă la faptul că singurul număr prim par este 2. Rezultă că vom verifica doar numere impare. Dar un număr impar poate avea doar divizori numere impare, deci d va crește cu 2 pornind de la valoarea 3. De asemenea, d nu va crește decât până la radical din n. Structura repetitivă va fi cu număr necunoscut de pași, astfel în momentul în care se stabilește că numărul nu este prim, se oprește verificarea.

```
citește n
dacă rest[n/2]=0
  atunci
    dacă n=2 atunci prim ← true
             altfel prim ← false
  altfel
    d ← 3
    prim ← true
    rad ← [√n ]
    cât timp d ≤ rad and prim execută
      dacă n mod d = 0
        atunci prim ← false
        altfel d ← d + 2
      sfârșit dacă
    sfârșit cât timp
sfârșit dacă
dacă prim atunci scrie 'Prim'
         altfel scrie 'Nu e prim'
```

Subiectul 2

1. Scrieți un subprogram care verifică dacă un șir de numere reale formează sau nu o progresie aritmetică. Subprogramul returnează o valoare logică corespunzătoare. Șirul este declarat ca variabilă globală.

Soluție

Varianta Pascal:

Presupunem că avem următoarea declarație de tip:

```
type sir=array[1..10] of Real;
```

Listing: II_1.PAS

```
function progresie(s:sir;dim:Integer)
                                  :Boolean;
var pr:Boolean;      { variabila auxiliara }
    ratia:Real;      {   diferenta dintre }
    i:Byte;        { primele doua elemente }
begin
  pr:=true; ratia:=s[2]-s[1]; i:=3;
  while (i<=dim) and pr do
    if s[i]-s[i-1]<>ratia then pr:=false
                          else Inc(i);
  progresie:=pr
end;
```

Varianta C:

Listing: PII_1.CPP

```
typedef int sir[10]

int progresie(sir s, int dim){
  int ratia=s[1]-s[0];
    // diferenta intre primele doua elemente
  for (int i=2;i<dim;i++)
    if (s[i]-s[i-1]!=ratio) return 0;
  return 1;
}
```

2. Fișierul text **DATE.TXT** conține numere scrise pe mai multe linii. Numărul de linii și numărul de numere pe linii nu se cunoaște. Scrieți un subprogram care creează un fișier nou în care sunt scrise numerele aflate la începutul fiecărei linii. Subprogramul va conține toate instrucțiunile necesare lucrului cu cele două fișiere.

Soluție

Varianta Pascal:

Listing: PII_2.PAS

```
procedure numere;
var fi,fo:Text;
    nr:Integer;
begin
  Assign(fi,'DATE.TXT'); Reset(fi);
  Assign(fo,'NR.TXT'); Rewrite(fo);
  while not Eof(fi) do
  begin
    Readln(fi,nr); { peste restul numerelor }
    Writeln(fo,nr) { de pe linie se "sare" }
  end;
  Close(fi); Close(fo)
end;
```

Varianta C:

Listing: PII_2.CPP

```
void numere(void){
  FILE *fi=fopen("DATE.TXT","rt"),
       *fo=fopen("NR.TXT","wt");
  char s[1000];
  int nr;
  while (!feof(fi)){
  fgets(s,1000,fi);
                    // peste restul numerelor
  sscanf(s,"%d",&nr);
                      // de pe linie se sare
    fprintf(fo,"%d\n",nr);
  }
  fclose(fi);
  fclose(fo);
}
```

Subiectul 3

Se consideră n mărgele de m culori (n≤10, m≤n), nu neapărat distincte. Să se scrie un program, folosind metoda backtracking, pentru afișarea tuturor posibilităților în care mărgelele se pot înșira astfel încât să nu existe două mărgele alăturate, de aceeași culoare.

Soluție

Varianta Pascal:

Listing MARGELE.PAS

```
var sirag:array[0..10] of Byte;  { solutia }
    culori:array[0..10] of string;
    pus:array[0..10] of Boolean;
      { vector cu valori booleene pentru a }
      { verifica, daca o anumita margea a  }
      { fost pusa deja sau nu in sirag     }
    i,nr_sol,n:Byte;

procedure citeste;
var i:Byte;
begin
  Write('Nr. margele:'); Readln(n);
  Writeln('Culorile margelelor, in ordine:');
  for i:=1 to n do Readln(culori[i])
end;

procedure scrie;
var j:Byte;
begin
  Inc(nr_sol);
  Writeln('Solutia ',nr_sol);
  for j:=1 to n do Writeln(culori[sirag[j]]);
end;

function ok(i:Byte):Boolean;
       { se verifica cerinta din problema }
begin
  ok:=true;
  if culori[sirag[i]]=culori[sirag[i-1]]
   { pentru margeaua curenta si precedenta }
  then ok:=false;
  if i=n
  then            { pentru ultima si prima }
    if culori[sirag[n]]=culori[sirag[1]]
    then ok:=false
end;

procedure insira(i:Byte);
var j:Byte;
begin
  for j:=1 to n do      { alegeri posibile }
  if not pus[j] then    { daca margeaua    }
     { respectiva nu este pusa inca in sirag}
  begin
    sirag[i]:=j;
    pus[j]:=true; { se pune in sirag }
    if ok(i)
    then if i<n then insira(i+1)
                else scrie;
    pus[j]:=false     { se scoate din sirag}
  end
end;

Begin
  citeste;
  nr_sol:=0;
  culori[0]:='';          { o margea fictiva }
  for i:=0 to n do pus[i]:=false;
     { nici o margea nu este inca in sirag }
  insira(1);
  if nr_sol=0
  then Writeln('Nu exista solutie!')
End.
```

Varianta C:

Listing MARGELE.CPP

```
#include <stdio.h>
#include <string.h>

int sirag[11], pus[11],i,nr_sol,n;
char culori[11][11];

void citeste(void){
  printf("Nr. margele:");
  scanf("%d",&n);
 printf("Culorile margelelor, in ordine:\n");
  for (int i=1;i<=n;i++)
    scanf("%s",culori[i]);
}

void scrie(void){
  printf("Solutia %d\n",++nr_sol);
  for (int j=1;j<=n;j++)
    printf("%s\n",culori[sirag[j]]);
}

int ok(int i){
        // se verifica cerinta din problema
  if (!strcmp(culori[sirag[i]],
                        culori[sirag[i-1]]))
    return 0;
   // pentru margeaua curenta si precedenta
  if (i==n && !strcmp(culori[sirag[n]],
              culori[sirag[1]])) return 0;
                 // pentru ultima si prima
  return 1;
}
```

```
void insira(int i){
  for (int j=1;j<=n;j++) // alegeri posibile
    if (!pus[j]){ // daca margeaua respectiva
                  // nu este pusa inca in sirag
     sirag[i]=j;
     pus[j]=1;              // se pune in sirag
     if (ok(i))
       if (i<n) insira(i+1);
       else scrie();
     pus[j]=0;          // se scoate din sirag
     }
}

void main(void){
  citeste();
  nr_sol=0;
  strcpy(culori[0],"");   // o margea fictiva
  for (int i=0;i<n;i++) pus[i]=0;
          // nici o margea nu este inca in
sirag
  insira(1);
  if (!nr_sol) puts("Nu exista solutie!");
}
```

Subiectul 4

Se consideră mediile semestriale la informatică dintr-o clasă. Datele cuprind nume, prenume și medie. Din aceste date se va crea o listă ordonată alfabetic, după nume. Parcurgeți lista în scopul calculării și afișării mediei generale a clasei, știind că mediile corigenților nu intră în media generală. Afișați numele corigenților.

Soluție:

Varianta Pascal:

Listing MEDII.PAS

```
Program medii_sem;
type lista=^elev;
     elev=record
            nume,pren:string[20];
            medie:Real;
            leg:lista
          end;
var cap,ult:lista;
    i,n:Byte;

procedure primul(var cap,ult:lista);
begin
  New(cap);
  Write('Numele: '); Readln(cap^.nume);
  Write('Prenumele: '); Readln(cap^.pren);
  Write('Media: '); Readln(cap^.medie);
  cap^.leg:=nil;
  ult:=cap
end;
```

```
procedure inserare(var cap,ult:lista);
var p,q:lista;
    gasit:Boolean;
    num,prenum,numtot:string[20];
begin
  gasit:=false;
  Write('Numele: '); Readln(num);
  Write('Prenumele: '); Readln(prenum);
  numtot:=num+prenum;
  p:=cap;
  while not gasit and (p<>nil) do
    if numtot>p^.nume+p^.pren
    then p:=p^.leg
    else gasit:=true;
  if gasit
  then {inserare}
  begin
    New(q);
    q^:=p^;
    p^.nume:=num;
    p^.pren:=prenum;
    Write('Media: '); Readln(p^.medie);
    p^.leg:=q;
    if p=ult then ult:=q;
  end
  else {ultimul}
  begin
    New(q);
    q^.nume:=num;
    q^.pren:=prenum;
    Write('Media: ');
    Readln(q^.medie);
    q^.leg:=nil;
    ult^.leg:=q;
    ult:=q
  end
end;

procedure calcul(cap:lista);
var med_gen:Real;
    n:Byte;
    exista_corigent:Boolean;
    p:lista;
begin
  p:=cap;
  n:=0;
  med_gen:=0;
  exista_corigent:=false;
  while p<>nil do
  begin
    if p^.medie>=5
    then
    begin
      med_gen:=med_gen+p^.medie;
      Inc(n)
    end
```

```
    else
    begin
      exista_corigent:=true;
      Writeln(p^.nume,' ',p^.pren,
                          ' e corigent.')
    end;
    p:=p^.leg
  end;
  if n=0 then
    Writeln('Toti elevii sunt corigenti!')
  else
   Writeln('Media generala: ',med_gen/n:6:3);
  if not exista_corigent
  then Writeln('Nu sunt corigenti!')
end;

Begin
  cap:=nil;
  Writeln('Nr. elevi in clasa: ');
  Readln(n);
  primul(cap,ult);
  for i:=2 to n do inserare(cap,ult);
  calcul(cap)
End.
```

Varianta C:

Listing MEDII.CPP

```
#include <stdio.h>
#include <string.h>

struct lista{
  char nume[20],pren[20];
  float medie;
  lista *leg;
} *cap, *ult;

int i,n;

void primul(lista* &cap, lista* &ult){
  cap = new lista;
  char c; // pentru citirea caracterului
          // de sfarsit de linie
  printf("Numele: "); gets(cap->nume);
  printf("Prenumele: "); gets(cap->pren);
  printf("Media: ");
  scanf("%f%c",&cap->medie,&c);
  cap->leg=NULL; ult-cap;
}

void inserare(lista* &cap, lista* &ult){
  lista *p, *q;
  char num[20],prenum[20],numtot[20],s[20];
  int gasit=0;
  char c; // pentru citirea caracterului
          // de sfarsit de linie
  printf("Numele: "); gets(num);
  printf("Prenumele: "); gets(prenum);
  strcpy(numtot, num);
  strcat(numtot, prenum);
  p=cap;
  while (!gasit && p){
    strcpy(s,p->nume);
    strcat(s,p->pren);
    if (strcmp(numtot,s)>0) p=p->leg;
    else gasit=1;
  }
  if (gasit){                          // inserare
    q=new lista;
    *q=*p;
    strcpy(p->nume,num);
    strcpy(p->pren,prenum);
    printf("Media: ");
    scanf("%f%c",&p->medie,&c);
    p->leg=q;
    if (p==ult) ult=q;
  }
  else{                                // ultimul
    q=new lista;
    strcpy(q->nume,num);
    strcpy(q->pren,prenum);
    printf("Media: ");
    scanf("%f%c",&q->medie,&c);
    q->leg=NULL;
    ult->leg=q;
    ult=q;
  }
}

void calcul(lista *cap){
  lista *p=cap;
  int n=0, exista_corigent=0;
  float med_gen=0;
  while (p){
    if (p->medie>=5){
      med_gen+=p->medie;
      n++;
    }
    else{
      exista_corigent=1;
      printf("%s %s este corigent.\n",
                          p->nume,p->pren);
    }
    p=p->leg;
  }
  if (!n) puts("Toti elevii sunt corigenti!");
  else printf("Media generala: %6.3f\n",
                                med_gen/n);
  if (!exista_corigent)
    puts("Nu sunt corigenti!");
}
```

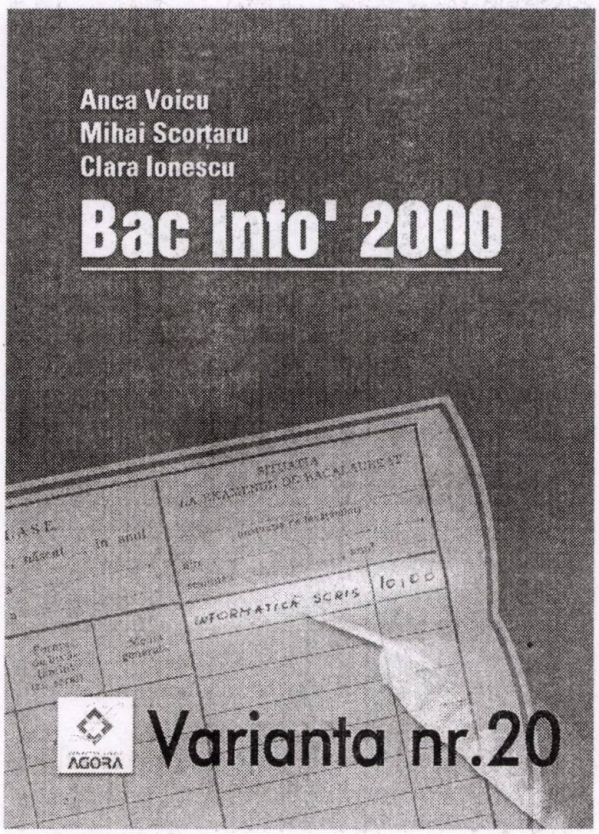
```
void main(void){
  cap=NULL;
  printf("Nr. elevi in clasa: ");
  char c;     // pentru citirea caracterului
              // de sfarsit de linie
  scanf("%d%c",&n,&c);
  primul(cap,ult);
  for (int i=1;i<n;i++) inserare(cap,ult);
  calcul(cap);
}
```

Barem

Subiectul 1: 30 puncte
1) 7,5 puncte
2) 7,5 puncte
3) 7,5 puncte
4) 7,5 puncte
Subiectul 2: 20 puncte
1) 10 puncte
2) 10 puncte
Subiectul 3: 20 puncte
Subiectul 4: 20 puncte

Se acordă 10 puncte din oficiu.

TOTAL: 100 puncte



news...

AMD Duron

Pe data de 27 aprilie Jerry Sanders, Chairman și CEO la AMD a anunțat denumirea viitorului procesor destinat calculatoarelor ieftine. Cunoscut până acum sub numele de cod Spitfire, noul procesor va intra pe piață purtând numele Duron, denumire derivată din verbul latin "durare" (a dura).

Începând din 6 iunie, acest procesor va concura direct cu Intel Celeron. Reamintim că Intel a lansat de curând o nouă generație de procesoare Celeron bazate pe miezul "Coppermine".

AMD Duron este bazat pe tehnologia Athlon, dar va avea cache L2 integrat pe chip, la fel ca și Celeron.

Se pare că Duron va oferi performanțe deosebite, foarte apropiate de ceea ce oferă actuala generație de procesoare AMD Athlon. Deoarece Duron depășește fără probleme noile procesoare Celeron în toate benchmark-urile, se poate trage concluzia că Duron va fi mult superior ofertei similare de la Intel.

Noul procesor nu va mai fi livrat în forma de Slot A, ci într-o nouă formă numită Socket A, deci va fi incompatibil cu plăcile de bază pentru actualele procesoare Athlon.

Inițial procesoarele Duron vor avea frecvențe cuprinse între 550 și 700 MHz și vor costa între 75$ și 175$.

Intel Pentium III 933

Intel a reușit să umple golul creat prin lansarea "forțată" a procesorului Pentium III 1GHz anunțând, în cursul lunii mai, disponibilitatea procesoarelor Pentium III la 933 MHz, după ce pe data de 20 martie a lansat procesoare Pentium III la 850 și 866 MHz.

Intel a reușit din nou să surprindă pe toată lumea prin prețul acestei noi variante de Pentium III. Un Pentium III la 1 GHz este cu aproximativ 30% mai ieftin decât unul la 933 MHz.

Se pare că Intel nu reușește să producă suficiente procesoare Pentium III la frecvențe mari, majoritatea companiilor fiind nevoite să ia în considerare ofertele de la AMD.

Introducere în teoria GRAFURILOR

Cristian Cadar

În acest ultim episod vom aborda o serie de algoritmi legați de determinarea fluxului maxim într-o rețea de transport: vom prezenta o serie de proprietăți ale fluxurilor, metoda Ford-Fulkerson de determinare a fluxului maxim și a fluxului maxim de cost minim, implementarea Edmonds-Karp a acestei metode și o serie de cazuri particulare de flux, cum este problema cuplajului maxim într-un graf bipartit.

Definiție

Printr-o ***rețea de transport*** înțelegem un graf orientat care conține două vârfuri speciale numite ***sursă*** și ***destinație*** și care are proprietatea că orice nod al său face parte dintr-un drum de la sursă la destinație.

Sursa unei rețele de transport se notează cu *s*, iar destinația cu *t*.

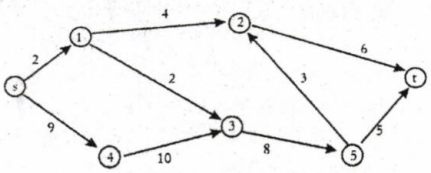


Figura 1: O rețea de transport

Exemplul clasic de rețea de transport este reprezentat de o serie de conducte dispuse între un robinet *s* și un canal de scurgere *t*. Fiecare conductă *(i, j)* este caracterizată prin cantitatea maximă de apă *c(i,j)*, care poate să treacă la un moment dat prin conductă. Problema aflării fluxului maxim presupune determinarea cantității maxime de apă care poate fi pompată prin robinetul *s* astfel încât pe nici o conductă să nu se depășească capacitatea maximă permisă. În figura 1 este prezentată o rețea de transport cu 7 noduri, dintre care o sursă *s* și o destinație *t*. Fiecărui arc îi este atașată o valoare pozitivă, reprezentând capacitatea maximă admisă pe arcul respectiv.

Pentru a rezolva problema fluxului maxim folosim algoritmul *Ford-Fulkerson*. Pentru aceasta dublăm arcele grafului, introducând un arc de sens contrar între oricare două noduri între care deja există un arc. Vom numi aceste arce introduse de algoritm *arce speciale*.

În fiecare moment suma celor două arce care unesc două noduri *i, j* ale grafului este egală cu capacitatea maximă dintre cele două vârfuri *c(i,j)*. Vom nota cu *a(i,j)* ponderea arcului care există inițial în graf și cu *a(j,i)* ponderea arcului special.

În fiecare moment al algoritmului, *a(i,j)* va reprezenta cantitatea de apă care mai poate fi pompată între nodurile *i* și *j*, iar *a(j,i)* va reprezenta valoarea fluxului dintre *i* și *j*. Deci în orice moment avem $a(i,j) + a(j,i) = c(i,j)$.

Iată, de exemplu, rezolvarea problemei fluxului maxim pentru graful din figura 1:

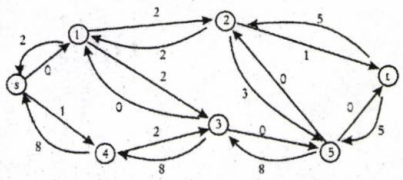


Figura 2: Fluxul maxim pentru rețeaua de transport din figura1

Avem de-a face deci cu un flux de valoare 10, fluxul pe fiecare arc fiind: pe arcul (*s*-4) un flux de valoare 8, pe arcul (*s*-1) un flux de valoare 2, pe arcul (1-3) un flux de valoare 0, pe arcul (5-*t*) un flux de valoare 5 etc.

De fapt, problema fluxului maxim cere găsirea unei funcții $f:U\to R$, (*U* este mulțimea arcelor), *f(i,j)* reprezentând valoarea fluxului pe arcul *(i,j)*. Funcția *f* trebuie să satisfacă următoarele proprietăți:

1) $f(i,j) \le c(i,j)$ pentru oricare două noduri *i, j*;

2) $\sum_i f(k,i) = \sum_j f(j,k)$ pentru orice nod $k \neq s, t$;

3) $\sum_i f(s,i) = \sum_j f(j,t) = F$, unde *F* este valoarea fluxului maxim.

Definiție:
Prin ***drum de creștere*** într-o rețea de transport înțelegem un drum de la *s* la *t* care conține arce de cost pozitiv, care pornesc atât din cele inițiale, cât și din cele speciale.

În figura 1, în care toate arcele speciale au ponderea 0, un posibil drum de creștere este drumul *s*-1-3-5-2-*t*. În figura 2, nu mai există nici un drum de creștere.

Valoarea unui drum de creștere este egală cu valoarea arcului de cost minim de pe drumul respectiv, numit arc critic. De exemplu, pentru drumul *s*-1-3-5-2-*t* din figura 1, valoarea acestuia este *min*(2, 2, 8, 3, 6) = 2 și avem două arce critice: (*s*-1) și (1-3).

Algoritmul *Ford-Fulkerson* are următorii pași:

```
Ford_Fulkerson(rețea transport G)
F←0
cât timp mai există drumuri de creștere
   determină un drum de creștere D
   v←valoarea lui D
   F←F+v
   pentru fiecare (i,j)∈D
      a(i,j)←a(i,j)-v
      a(j,i)←a(j,i)+v
```

În cele ce urmează vom încerca să lămurim cititorul de corectitudinea acestui algoritm: să presupunem că pentru graful din figura 1 găsim la început drumul *s*-1-3-5-*t*, la pasul 2 drumul *s*-4-3-5-*t* și la pasul 3 drumul *s*-4-3-5-2-*t*. Va rezulta graful din figura 3:

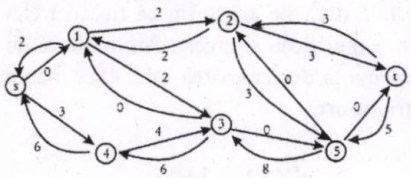


Figura 3

Aceste prime trei drumuri sunt compuse doar din arce care existau inițial în graf. În acest moment avem un flux de valoare 8 și nu mai există drumuri de creștere formate doar din arce inițiale. Să vedem ce semnificație au drumurile care conțin și arce speciale. În cazul de față singurul drum de creștere care mai există în graf este drumul de valoare 2: *s*-4-3-1-2-*t* care conține arcul special (3-1). Semnificația parcurgerii acestui arc este că "scoatem" apa de pe conducta respectivă. Studiind acum ce se întâmplă cu trecerea apei prin conducte, se remarcă următoarele:

- pe conducta (1-3) am scos toată apa, deci pe aici nu mai trece nici o cantitate de apă;
- în nodul 1 am redirecționat fluxul de valoare 2 (care ajungea prin conducta (*s*-1)) prin conducta (1-2);
- fluxul care ajunge în nodul 3 va fi în continuare de valoare 8, numai că acum va ajunge în întregime de pe conducta (4-3), deoarece pe conducta (1-3) s-a scos toată apa.

Să analizăm complexitatea algoritmului *Ford-Fulkerson*: găsirea unui drum de creștere se face în $O(n+m)$, la fiecare iterație valoarea fluxului este mărită cu cel puțin o unitate, deci numărul maxim de iterații este F. În concluzie, complexitatea algoritmului *Ford-Fulkerson* este $O(F(n+m))$. Pentru valori mari ale fluxului, această dependență de valoarea fluxului poate deranja. Să considerăm graful din figura 4. În cazul în care algoritmul va găsi drumurile de creștere *s*-1-*t* și *s*-2-*t*, algoritmul se va termina în două iterații. În cazul în care algoritmul va găsi alternativ drumurile *s*-1-2-*t* și *s*-2-1-*t*, acesta se va termina după 2000000 de iterații!

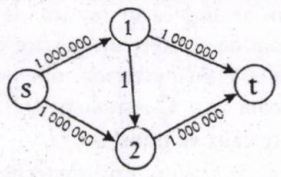


Figura 4

Pentru a evita astfel de situații vom prezenta succint ideea algoritmului *Edmonds-Karp*, care este de fapt o implementare eficientă a algoritmului *Ford-Fulkerson*. Acest algoritm, în loc să găsească un drum de creștere la întâmplare, va găsi la fiecare pas un drum de creștere cu număr minim de arce. Acest lucru se realizează foarte ușor, printr-o simplă parcurgere în lățime. Astfel, se garantează că numărul maxim de iterații pe care le realizează algoritmul este $n \cdot m$. Demonstrația acestui fapt este destul de simplă: fiecare drum de creștere conține, așa cum rezultă din definiție, cel puțin un arc critic. Să presupunem că arcul *(i,j)* apare pentru prima oară ca arc critic într-un drum de creștere în momentul în care acesta se află la distanța d de sursă.

Se arată ușor că, alegând de fiecare dată un drum de creștere cu număr minim de muchii, în momentul în care arcul *(i,j)* va apare pentru a doua oară ca arc critic, el se va afla la o distanță de cel puțin $d+2$ de sursă.

În concluzie fiecare arc *(i-j)* va putea fi critic de cel mult $[n/2]$ ori, deci vom avea cel mult $O(n)$ drumuri de creștere caracterizate prin arcul critic *(i-j)*. În total vor exista cel mult $O(n \cdot m)$ drumuri de creștere, deci complexitatea totală a implementării algoritmului *Edmonds-Karp* va fi $O((n+m) \cdot n \cdot m) = O(n \cdot m)^2$. Un alt fapt interesant este găsirea tăieturii de capacitate minimă.

Definiție:
Fie $G(V,U)$ o rețea de transport și $S \subset V$ o mulțime de noduri astfel încât $s \in S$ și $t \notin S$. Mulțimea arcelor *(i,j)* cu *i* în S și *j* în $V \setminus S$ se numește *tăietură în rețeaua de transport G*. Capacitatea unei tăieturi este dată de suma capacităților arcelor care o compun. Tăietura se notează cu S, iar capacitatea acesteia cu $c(S)$.

Folosindu-ne de exemplul dat anterior, problema găsirea tăieturii de capacitate minimă presupune găsirea acelor conducte pentru care suma capacităților este minimă și prin "astuparea" cărora apa nu mai poate ajunge de la sursă la destinație.

În cele ce urmează vom prezenta *teorema de flux maxim - capacitate minimă* care demonstrează corectitudinea algoritmului *Ford-Fulkerson* și care sugerează algoritmul de găsire a tăieturii de capacitate minimă.

Teoremă:
Fie o rețea de transport $G(V,U)$ și F valoarea fluxului maxim din rețea. Atunci următoarele afirmații sunt echivalente:
1) F este flux maxim în rețea
2) Rețeaua de transport nu conține nici un drum de creștere
3) Există o tăietură S pentru care $F=c(S)$
1) ⇒ 2) Această implicație este evidentă. Să presupunem prin absurd că în rețeaua de transport am mai găsi un drum de creștere de valoare v. Atunci, conform algoritmului *Ford-Fulkerson*, am putea mări valoarea fluxului maxim cu v. Contradicție cu faptul că F este fluxul maxim în rețeaua de transport G.
2) ⇒ 3) Dacă rețeaua de transport nu conține drumuri de creștere, atunci vom considera mulțimea S ca fiind mulțimea tuturor vârfurilor la care se mai poate ajunge din s. Evident t nu aparține lui S - deoarece în G nu mai există drumuri de creștere - deci S este o tăietură validă. Remarcăm că pentru fiecare arc (i,j) din S ($i \in S$ și $j \in V \setminus S$) $f(i,j) = c(i,j)$ deoarece altfel am fi putut ajunge și în nodul j, deci j ar fi făcut parte din S (contradicție). Dar:

$$\sum_{(i,j)\in S_+} f(i,j) = F \text{, deci } c(S)=F.$$

3) ⇒ 1) Este evident că pentru o tăietură oarecare X din G, fluxul maxim $F \le c(X)$. Atunci, din egalitatea $F=c(S)$ rezultă că F este fluxul maxim din rețeaua de transport G.

Teorema indică și algoritmul de determinare a tăieturii de capacitate minimă într-o rețea de transport: se aplică algoritmul *Ford-Fulkerson* și se reține nodurile la care putem ajunge din s în momentul în care nu mai avem drumuri de creștere. Aceste noduri vor forma mulțimea S care caracterizează tăietura de capacitate minimă. Așa cum am demonstrat în teorema de mai sus, valoarea tăieturii de capacitate minimă este egală cu valoarea fluxului maxim.

În momentul în care într-o rețea de transport avem asociat pentru fiecare arc, pe lângă capacitate, și un cost unitar, atunci se pune problema aflării unui flux maxim de cost minim. Această problemă se poate rezolva tot cu ajutorul algoritmului *Ford-Fulkerson*, cu condiția ca la fiecare pas să se determine un drum de creștere de cost minim. Aflarea unui astfel de drum este posibilă folosind algoritmul *Bellman-Ford*. Algoritmul *Dijkstra* nu se poate aplica în acest caz, deoarece - așa cum am arătat într-un articol precedent - acest algoritm nu funcționează în cazul existenței muchiilor negative, iar în acest caz costul arcelor introduse de algoritm va putea fi negativ.

Alte situații speciale care pot apărea sunt:

- *rețea de transport cu mai multe surse și mai multe destinații*: în acest caz introducem o *supersursă ss* și o *superdestinație st*. Ducem de la *ss* la toate sursele arce de capacitate și de asemenea, de la toate destinațiile la *st* arce de capacitate. În acest fel am redus problema la determinarea fluxului într-o rețea de transport cu o singură sursă *ss* și o singură destinație *st*;
- *rețea de transport cu capacități și în noduri*, adică într-un nod poate intra o cantitate de apă cel mult egală cu capacitatea nodului respectiv. În acest caz, desfacem fiecare nod n în două noduri $n1$ și $n2$, între care punem un arc de capacitate egală cu capacitatea nodului n. Toate arcele care intrau în nodul inițial n vor intra acum în nodul $n1$, iar toate arcele care ieșeau din nodul inițial n vor ieși acum din nodul $n2$. Astfel, am redus problema la determinarea fluxului maxim într-o rețea de transport cu capacități doar pe arce;
- *problema cuplajului maxim într-un graf bipartit* este un caz particular de flux maxim într-o rețea de transport. Să presupunem că avem o listă de n elevi și n licee și o listă de preferințe de forma (e, l) având semnificația că elevul e dorește să studieze la liceul l. Fiecare elev poate să dorească în egală măsură să studieze la mai multe licee. Problema cere ca - dacă este posibil - să se atașeze fiecărui elev un liceu unic, astfel încât fiecare elev să fie repartizat într-un liceu în care dorește să studieze. Problema se modelează printr-un graf bipartit, având pe de o parte mulțimea elevilor, iar pe de cealaltă parte mulțimea liceelor. Între un elev și un liceu există arc doar dacă elevul dorește să studieze la liceul respectiv. Problema cere selectarea unui număr maxim de arce ale grafului bipartit, astfel încât fiecărui elev să-i fie asociat un liceu unic și invers. În cazul în care numărul maxim de perechi valide este n, problema are soluție. Pentru rezolvare, introducem o sursă virtuală s și o destinație virtuală t. Ducem arc între nodul sursă și fiecare elev, și între fiecare liceu și nodul destinație. Toate arcele din problemă vor avea capacitatea 1. Astfel, ne asigurăm că fiecărui elev îi este asociat un singur liceu și invers. Am redus și de această dată problema la determinarea unui flux maxim într-o rețea de transport.

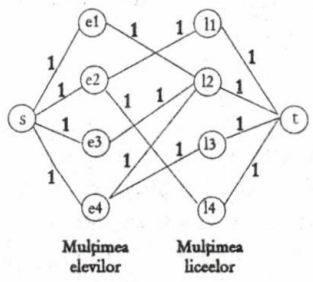


Figura 5: Problema cuplajului maxim pentru n=4

Bibliografie

1. Cormen T., Leiserson C., Rivest R., *Introducere în algoritmi*, Editura Computer Libris Agora, 2000
2. Georgescu H., Livovschi L., *Analiza și sinteza algoritmilor*, Editura Științifică și Enciclopedică, 1986
3. Mateescu E., Maxim I., *Arbori*, Editura Țara Fagilor, 1996
4. Cadar C., Stroe M., *Culegere de probleme și programe PASCAL*, Editura Petrion, 1999

Pagini WEB cu XML 1.0

Mihai Scorțaru

Prezentarea tehnologiei XML se încheie cu insulele de date, modalitatea prin care poate fi accesată dintr-o pagină Web informația stocată în format XML. De asemenea, vom prezenta câteva detalii referitoare la XSL, limbaj care permite manipularea datelor XML.

Insule de date

În cadrul episodului anterior am folosit deja o insulă de date pentru a exemplifica parcurgerea unui DOM. Vom prezenta acum detaliat modul în care pot fi folosite insulele de date.

În cadrul unei pagini Web o insulă de date poate fi creată cu ajutorul marcajului **<XML>**. Pentru a crea o insulă de date care preia informațiile dintr-un alt fișier veți putea folosi construcția:

<XML id=*nume_insulă* **src**=*nume_fișier*>.

În cazul în care doriți ca informațiile să nu fie preluate dintr-un alt fișier, ci să fie conținute chiar în documentul HTML respectiv, veți putea folosi o construcție de tipul:

```
<XML id=nume_insulă>
        informație în format XML
</XML>
```

Să presupunem că fișierul **AUTORI.XML** conține următoarele informații:

```
<AUTORI>
  <AUTOR>
    <ID>1001</ID>
    <NUME>Oltean</NUME>
    <PRENUME>Mihai</PRENUME>
  </AUTOR>
  <AUTOR>
    <ID>1002</ID>
    <NUME>Giumale</NUME>
    <PRENUME>Cristian</PRENUME>
  </AUTOR>
</AUTORI>
```

Dacă dorim să creăm o insulă de date folosind informațiile din acest fișier, marcajul folosit ar putea fi **<XML id**="insula" **src**="AUTORI.XML">. Vă prezentăm în continuare construcția echivalentă pentru cazul în care se dorește ca informațiile să fie păstrate în cadrul documentului HTML:

```
<XML id="insula">
  <AUTORI>
    <AUTOR>
      <ID>1001</ID>
      <NUME>Oltean</NUME>
      <PRENUME>Mihai</PRENUME>
    </AUTOR>
    <AUTOR>
      <ID>1002</ID>
      <NUME>Giumale</NUME>
      <PRENUME>Cristian</PRENUME>
    </AUTOR>
  </AUTORI>
</XML>
```

Pentru a stabili o relație între elementele unei pagini Web și anumite date în format XML avem două posibilități: fie stabilim o legătură pentru un singur element, fie stabilim o legătură pentru un tabel și apoi legături pentru elementele tabelului.

O legătură pentru un singur element poate fi realizată astfel:

```
<INPUT type="text" datasrc="#insula"
       datafld="NUME">
</INPUT>.
```

Vă prezentăm în continuare un document HTML care folosește această tehnică de realizare a legăturii:

```
<HTML>
  <HEAD>
    <TITLE>Legătură simplă</TITLE>
  </HEAD>
  <BODY>
    <XML id="insula">
      <AUTORI>
        <AUTOR>
          <ID>1001</ID>
          <NUME>Oltean</NUME>
```

```
        <PRENUME>Mihai</PRENUME>
      </AUTOR>
      <AUTOR>
        <ID>1002</ID>
        <NUME>Giumale</NUME>
        <PRENUME>Cristian</PRENUME>
      </AUTOR>
    </AUTORI>
  </XML>
  <FONT size="5">
    Nume: <INPUT type="text" size="10"
                 datasrc="#insula"
                 datafld="NUME"
                 readonly></INPUT>
    Prenume: <INPUT type="text"
                    size="10"
                    datasrc="#insula"
                    datafld="PRENUME"
                    readonly></INPUT>
  </FONT>
 </BODY>
</HTML>
```

Pagina Web corespunzătoare acestui document HTML este prezentată în figura 1.

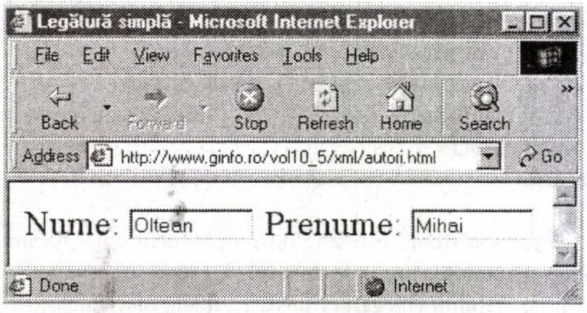


Figura 1: Legătură pentru un singur element

Pentru a realiza o legătură pentru un tabel, construcția folosită este următoarea:

```
<TABLE datasrc="#insula">
  <TR>
    <TD><INPUT type="text"
               datafld="nume">
        </INPUT>
    </TD>
    <TD><INPUT type="text"
               datafld="nume">
        </INPUT>
    </TD>
  </TR>
</TABLE>
```

Un document HTML care folosește această tehnică este prezentat în continuare:

```
<HTML>
  <HEAD>
    <TITLE>Legãturã pentru tabel</TITLE>
  </HEAD>
  <BODY>
    <XML id="insula">
      <AUTORI>
        <AUTOR>
          <ID>1001</ID>
          <NUME>Oltean</NUME>
          <PRENUME>Mihai</PRENUME>
        </AUTOR>
        <AUTOR>
          <ID>1002</ID>
          <NUME>Giumale</NUME>
          <PRENUME>Cristian</PRENUME>
        </AUTOR>
      </AUTORI>
    </XML>
    <DIV align="center">
      <B>Autori</B>
    </DIV>
    <HR width="90%" align="center">
    <TABLE cellspacing="10"
           datasrc="#insula">
      <TR>
        <TD><INPUT type="text" size="10"
                   datafld="NUME"
                   readonly>
            </INPUT>
        </TD>
        <TD><INPUT type="text" size="10"
                   datafld="PRENUME"
                   readonly>
            </INPUT>
        </TD>
      </TR>
    </TABLE>
    <HR align="center">
    &copy; 2000
    <B>theGuild International</B>
  </BODY>
</HTML>
```

Pagina Web corespunzătoare acestui document HTML este prezentată în figura 2.

În cadrul exemplului anterior am folosit o structură de date în cadrul căreia, practic, informațiile erau structurate pe doar două niveluri, primul corespunzător autorului, iar al doilea corespunzător informațiilor despre autor. Din acest motiv am putut foarte ușor să asociem liniile tabelului unui autor și coloanele diferitelor informații despre autor. Problema apare în cazul în care avem o ierarhie cu mai multe niveluri; în acest caz o soluție este imbricarea tabelelor. Vă prezentăm în continuare un exemplu în care este folosită această metodă:

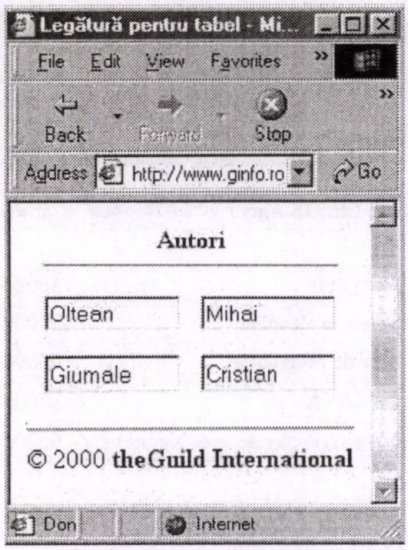


Figura 2: Legătură pentru un tabel

```
<HTML>
  <HEAD>
    <TITLE>Computer LIBRIS Agora</TITLE>
  </HEAD>
  <BODY>
    <XML id="insula">
      <EDITURI>
        <EDITURA>
          <DENUMIRE>
            Computer LIBRIS Agora
          </DENUMIRE>
          <ORAS>Cluj-Napoca</ORAS>
          <JUDET>Cluj</JUDET>
          <TARA>Romania</TARA>
          <CARTE>
            <TITLU>
              Bac Info&amp;apos; 2000
            </TITLU>
            <PRET>30000</PRET>
          </CARTE>
          <CARTE>
            <TITLU>
              Introducere in algoritmi
            </TITLU>
            <PRET>330000</PRET>
          </CARTE>
          <CARTE>
            <TITLU>
              Un atelier de programare
            </TITLU>
            <PRET>100000</PRET>
          </CARTE>
          <CARTE>
            <TITLU>Limbajul JAVA</TITLU>
            <PRET>80000</PRET>
          </CARTE>
          <CARTE>
            <TITLU>
              Proiectarea si
              implementarea algoritmilor
            </TITLU>
            <PRET>65000</PRET>
          </CARTE>
          <ANGAJAT>
            <NUME>Ionescu</NUME>
            <PRENUME>Clara</PRENUME>
            <FUNCTIE>
              Director general
            </FUNCTIE>
          </ANGAJAT>
          <ANGAJAT>
            <NUME>Ionescu</NUME>
            <PRENUME>Eugen</PRENUME>
            <FUNCTIE>
              Director executiv
            </FUNCTIE>
          </ANGAJAT>
          <ANGAJAT>
            <NUME>Bucur</NUME>
            <PRENUME>Carmen</PRENUME>
            <FUNCTIE>
              Director comercial
            </FUNCTIE>
          </ANGAJAT>
        </EDITURA>
      </EDITURI>
    </XML>
    <TABLE id="tabel_edituri"
           datasrc="#insula" border="1">
      <TR>
        <TD><DIV datafld="DENUMIRE">
            </DIV></TD>
        <TD><DIV datafld="ORAS">
            </DIV></TD>
        <TD><DIV datafld="TARA">
            </DIV></TD>
      </TR>
      <TR>
        <TD colspan="4">
          <TABLE>
            <TR>
              <TD>Carti</TD>
            </TR>
            <TR>
              <TD colspan="4">
                <TABLE datasrc="#insula"
                       datafld="CARTE"
                       border="1">
                  <TR>
                    <TD>
                      <DIV datafld=
                           "TITLU">
```

```
                    </DIV>
                  </TD>
                  <TD align="right">
                    <DIV datafld=
                                "PRET">
                    </DIV>
                  </TD>
                </TR>
              </TABLE>
            </TD>
          </TR>
          <TR>
            <TD>Angajati</TD>
          </TR>
          <TR>
            <TD colspan="4">
              <TABLE datasrc="#insula"
                     datafld="ANGAJAT"
                     border="1">
                <TR>
                  <TD>
                    <DIV datafld=
                                "NUME">
                    </DIV>
                  </TD>
                  <TD>
                    <DIV datafld=
                                "PRENUME">
                    </DIV>
                  </TD>
                  <TD>
                    <DIV datafld=
                                "FUNCTIE">
                    </DIV>
                  </TD>
                </TR>
              </TABLE>
            </TD>
          </TR>
        </TABLE>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Pagina Web corespunzătoare documentului HTML anterior este prezentată în figura 3.

Stilizare folosind XSL

XSL este prescurtarea de la *eXtensible Styling Language* (limbaj extensibil de stilizare) și este un limbaj bazat pe XML care permite manipularea datelor păstrate în format XML. XSL poate transforma date dintr-un anumit format XML în alt format XML, dintr-un format XML în format HTML etc.

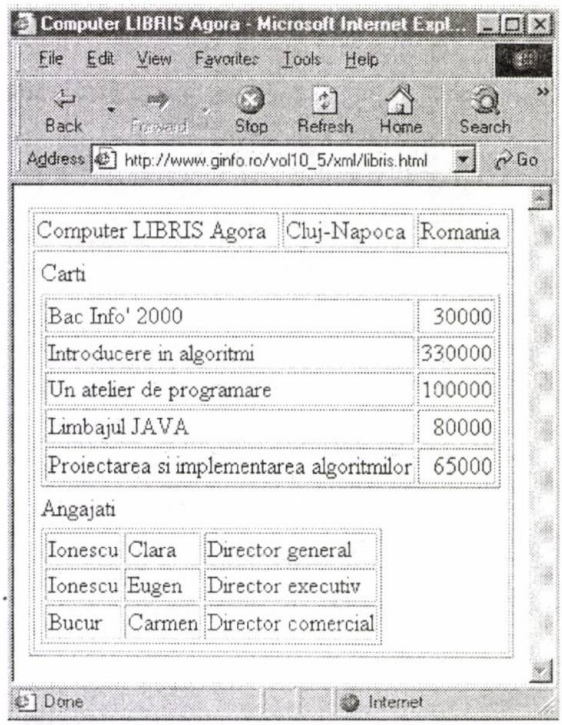


Figura 3: Legături cu tabele imbricate

Unul dintre motivele pentru care a apărut XSL este faptul că XML poate fi folosit numai pentru stocarea datelor. Am mai spus că marcajele XML ajută doar la identificarea datelor spre deosebire de marcajele HTML care descriu și modul în care datele sunt afișate în cadrul paginii Web. Din aceste motive, dacă dorim să afișăm date păstrate în format XML, va trebui să găsim o modalitate de a le "stiliza" (a descrie modul în care sunt afișate).

XSL este compus din două părți importante: un *limbaj de transformare* și un *limbaj de formatare*.

Modele de stiluri XSL

Ideea care stă la baza limbajului XSL este aceea că avem un set de reguli care sunt puse în corespondență cu elementele și atributele XML. Aceste reguli poartă denumirea de *șabloane* (engl. *templates*); în cadrul unui șablon se pot traversa elementele și atributele, se pot aplica alte șabloane și se pot efectua alte tipuri de prelucrări.

Orice text care nu face parte dintr-o instrucțiune de procesare XSL este considerată *ieșire* și aceasta este modalitatea de transformare a datelor XML: stabilirea de corespondențe cu elementele și furnizarea textului sau a anumitor valori la ieșire.

Pentru a exemplifica folosirea limbajului XSL să considerăm următorul document XML:

```
<xml xmlns:s=
          "identificator_unic_1_pentru_s"
     xmlns:rs=
          "identificator_unic_2_pentru_rs"
     xmlns:z="#Schema"
```

```
<s:Schema id="Schema">
  <s:ElementType name="row"
                 content="eltOnly">
    <s:AttributeType name="id"
                     rs:number="1"
                     rs:writeunknown=
                                 "true">
    ...
    </s:ElementType>
  </s:Schema>
  <rs:data>
    <z:row id="10001"
           nume="Scortaru"
           prenume="Mihai"
           telefon="40 92 327635"
           adresa="65 Teleorman Ap. 46"
           oras="Cluj-Napoca"
           judet="Cluj"
           tara="Romania"
           cod="3400"/>
    ...
  </rs:data>
</xml>
```

Dacă dorim să transformăm documentul XML anterior într-un document HTML folosind XSL, va trebui să creăm următorul document:

```
<xsl:stylesheet xmlns:xsl=
          "http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <BODY>
        <xsl:apply-templates
          select="//rs:data"/>
      </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match="//rs:data">
    <TABLE border="1">
      <xsl:for-each select="z:row">
        <TR>
          <xsl:for-each select="(@*)">
            <TD>
              <xsl:value-of/>
            </TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

În cele ce urmează vă vom explica conținutul documentului prezentat anterior. Pentru început trebuie înțeles faptul că XSL este în format XML, deci este format dintr-o serie de marcaje. Marcajul de pe primul nivel are, de obicei, forma:

<xsl:stylesheet xmlns:xsl="*identificator_unic***">**

Acest marcaj este folosit pentru a identifica modelul de stil și zona pentru nume corespunzătoare modelului respectiv. Deoarece totul este în format HTML, va trebui ca la sfârșit să avem marcajul de închidere

</xsl:stylesheet>.

În interiorul unui model de stil, marcajele de pe nivelul imediat următor corespund șabloanelor definite în cadrul modelului de stil.

Comanda **<xsl:template match="/">** este folosită pentru a defini primul șablon care este folosită pentru a realiza corespondența cu un anumit element, în acest caz elementul rădăcină care este simbolizat prin caracterul /.

În acest moment am început prelucrarea arborelui XML. Datorită faptului că dorim să realizăm o conversie în format HTML va trebui să furnizăm la ieșire marcajele specifice limbajului HTML. Din acest motiv, următoarele două linii sunt:

```
<HTML>
  <BODY>
```

Acum va trebui să găsim informațiile propriu-zise stocate prin intermediul elementului **<rs:data>**. Pentru a folosi aceste informații vom crea un alt șablon care va fi aplicat și după terminarea prelucrărilor corespunzătoare acestui șablon, se vor continua prelucrările din șablonul curent. Pentru a accesa un alt șablon comanda folosită este **<xsl:apply-templates select="//rs:data"/>**. După terminarea prelucrărilor corespunzătoare acestui șablon putem să "terminăm" documentul HTML introducând liniile:

```
  </BODY>
</HTML>
```

Secțiunea corespunzătoare primului șablon se încheie cu marcajul de închidere **</xsl:template>**.

Acum începe esența limbajului XSL și anume prelucrarea datelor. Vom începe prin a crea un șablon cu ajutorul marcajului de deschidere **<xsl:template match ="//rs:data">**. După cum se poate observa din documentul XML pe care l-am prezentat anterior toate datele sunt păstrate în cadrul elementului **<rs:data>** și fiecare linie este păstrată în cadrul unui element **<z:row>**. Câmpurile (informațiile propriu-zise) sunt păstrate prin intermediul atributelor acestui element.

Pentru crearea tabelului va trebui, pentru început, să furnizăm la ieșire marcajul HTML corespunzător:

<TABLE border="1">.

Va trebui acum să prelucrăm datele din fiecare linie, fiecărei astfel de linii corespunzându-i o linie în tabelul HTML. Pentru aceasta vom folosi "instrucțiunea" `for-each`, asemănătoare instrucțiunii `for..each` din limbajul *VBScript*. Aceasta este o instrucțiune repetitivă care ia în considerare fiecare element dintr-o anumită colecție de elemente. Pentru a preciza colecția din care sunt preluate elementele, se folosește atributul **select**. În cazul nostru

va trebui să parcurgem toate elementele `z:row`, deci "instrucțiunea" folosită va fi:

```
<xsl:for-each select="z:row">.
```

Trebuie să creăm linia de tabel corespunzătoare; pentru aceasta vom furniza la ieșire marcajul HTML **<TR>**.

Va trebui să prelucrăm acum toate atributele elementului `z:row`, acestea corespunzând câmpurilor noastre. Vom folosi din nou construcția `for-each`, dar de această dată valoarea atributului **select** va fi una specială. Va trebui să găsim fiecare atribut, indiferent de denumirea acestuia, deci vom folosi un asterisc pentru a indica acest lucru. Pentru a lua în considerare atribute va trebui să folosim caracterul `@`, deci pentru a găsi toate atributele construcția corectă este `@*`. În concluzie, construcția noastră este **<xsl:for-each select="(@*)">**.

Urmează să creăm celulele tabelului, deci vom furniza la ieșire marcajul HTML care realizează acest lucru: **<TD>**.

În interiorul celulei dorim să afișăm valoarea atributului. "Instrucțiunea" `value-of` indică faptul că va fi furnizată la ieșire valoarea unui nod și noi o vom folosi pentru a furniza valoarea nodului curent: **<xsl:value-of/>**.

Cam aceasta a fost prelucrarea datelor folosind XSL. Pentru a încheia, va trebui să închidem toate marcajele pe care le-am deschis, deci fișierul va mai conține liniile:

```
            </TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

Chiar dacă această prelucrare pare destul de dificilă la prima vedere, ea nu este foarte depărtată de programarea "adevărată". Limbajul XSL pare ciudat, deoarece specificațiile sale sunt în XML și sintaxa sa este bazată pe realizarea de corespondențe cu nodurile din documente XML. Totuși, construcțiile sunt destul de apropiate de standardele actuale în programare.

Cea mai convenabilă modalitate de a aplica un model de stiluri XSL este de a-l specifica în interiorul documentului XML. Pentru a realiza acest lucru, prima linie a documentului XML respectiv va trebui să fie de forma:

```
<?xml-stylesheet type="text/xsl"
    href="nume_fisier.xsl"?>
```

La final

Cam atât despre paginile Web cu XML 1.0. Este evident faptul că subiectul este departe de a fi epuizat, dar pentru a continua ar trebui să prezentăm detalii tehnice care ar fi interesante doar pentru un segment destul de restrâns al cititorilor.

Serialul despre crearea paginilor Web a fost unul dintre cele mai apreciate seriale apărute în GInfo, mulți cititori fiind interesați de modul în care sunt create paginile Web începând cu tehnicile simple cum ar fi limbajul HTML și continuând cu metode mai avansate cum ar fi limbajul *JavaScript*.

Cea mai mare problemă cu acest serial este că nu știm dacă este momentul să-i punem punct. Cei care trebuie să decidă în această privință nu pot fi decât cititorii noștri. Așadar, vă rugăm să ne spuneți dacă doriți ca acest serial să continue și, în cazul în care acest serial nu va dispărea, să precizați ce ați mai dori să aflați. Noi vă putem da doar câteva sugestii: XHTML (rescrierea limbajului HTML în format XML), SMIL (limbaj pentru prezentări multimedia), VRML (limbaj pentru crearea de realități virtuale), ASP (pagini Web pe server), *VBScript* (alternativă la *JavaScript*).

Mihai Scorțaru este student în anul II la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa smihai@asterix.obs.utcluj.ro.

Acces la Internet

Numărul utilizatorilor de servicii Internet la nivel mondial a trecut de 276 de milioane în 1999 și va crește cu aproape 100 de milioane în 2000. Majoritatea noilor utilizatori vor proveni din Asia; până în 2005 China va deveni a doua piață Internet din lume după SUA, din punctul de vedere al numărului utilizatorilor. La sfârșitul anului 2000, Statele Unite vor avea pe teritoriul propriu peste 36% din totalul mondial de utilizatori Internet, față de 46% cât reprezentau aceștia în 1998 și 56% în 1996. Aceste date au fost publicate în urma unui studiu de piață efectuat de compania eTForecasts.

Windows 2000 SP 1

Deși Microsoft spera ca tranziția de la Windows NT 4.0 la Windows 2000 să fie una rapidă, numeroase companii au preferat să nu riște să folosească acest nou sistem de operare pentru că nu au la dispoziție un prim pachet care să rezolve problemele care au fost deja raportate.

Pentru a remedia situația, Microsoft a anunțat că în curând vor începe testările beta pentru Windows 2000 Service Pack 1 (W2K SP1). Pachetul va rezolva problemele cunoscute ale tuturor versiunilor Windows 2000 disponibile pe piață: Professional, Server și Advanced Server.

POȘTA electronică și FTP

Marin Vlada

Încheiem prezentarea mediului Internet cu detalii referitoare la transmiterea și primirea de mesaje, folosind Outlook Express și Netscape Messanger. De asemenea, vom prezenta pe scurt și modul în care pot fi transferate fișiere, folosind protocolul FTP.

Sub sistemul de operare *Windows* există mai multe programe care realizează serviciul de *e-mail*, serviciu ce poate fi apelat prin intermediul programelor de navigare: *Microsoft Internet Explorer* și *Netscape Communicator*.

Față de programele ce lucrează sub sistemele *Linux/Unix*, sub sistemul *Windows* programele de *e-mail* oferă interfețe grafice foarte comode și editoare perfecționate pentru compunerea mesajelor, dar și programe perfecționate pentru codificarea și decodificarea fișierelor atașate unui mesaj. În prezent, aceste codificări și decodificări, de regulă, se realizează automat, fără ca utilizatorul să aibă grijă de acest lucru.

Comentariu

Programul de procesare texte și imagini *Microsoft Word '97* oferă prin meniul **File**, pe lângă altele și funcțiile pentru *Internet* (*e-mail*):

- **Send To | Mail Recipient**: trimiterea documentului procesat la un destinatar cu adresă de *e-mail*;
- **Save As | HTML Document**: salvarea documentului procesat ca text HTML, limbaj destinat rețelei WWW.

Programul *Word '97* oferă butoanele **Web Toolbar** și **Insert Hyperlink** pentru generarea de pagini Web destinate rețelei WWW.

Pentru utilizarea serviciului *e-mail* sub un program de navigare, cum ar fi *Internet Explorer* sau *Netscape Communicator*, utilizatorul trebuie să realizeze diverse setări (configurare de lucru) la prima utilizare. Cu această ocazie se poate alege programul care oferă serviciile proprii de poștă electronică, și anume:

- *Outlook Express*;
- *Netscape Messenger*;
- *Microsoft Outlook*;
- *Windows Messaging*.

În primul rând, după instalarea programului de navigare, trebuie să se opteze pentru preferințele de lucru.

Internet Explorer

În cazul programului de navigare *Internet Explorer* opțiunile de lucru pot fi stabilite folosind fereastra *Internet Options* (meniul **Tools** comanda **Internet Options**...) care oferă următoarele etichete (meniuri):

- *General*: *Home page*, *Temporary Internet Files*, *History*;
- *Security*: *zone*;
- *Content*: *Content Advisor*, *Certificates*, *Personal Information*;
- *Connection*: *Proxy Server*;
- *Programs*: *Messaging*, *Personal Information*;
- *Advanced*.

Outlook Express

Serviciile de *e-mail* sunt realizate de programul ales pentru acest lucru, de exemplu, în cazul nostru de programul *Outlook Express*.

Programul conține câteva foldere standard:

- **Inbox**: mesaje primite, ce nu sunt redirecționate către alte directoare;
- **Outbox**: mesaje ce urmează a fi trimise;
- **Sent Items**: mesaje trimise;
- **Deletes Items**: mesaje șterse (primite, trimise) și abandonate ulterior.

Serviciile oferite de *Outlook Express* sunt:

- *Read Mail*: citirea corespondenței (*e-mail*);
- *Read News*;
- *Compose a Message*: compunerea mesajelor și trimiterea acestora;
- *Address Book*: lista de adrese *e-mail* de care utilizatorul are nevoie frecvent;
- *Download All*;
- *Find People*.

Configurarea *Outlook* este realizată prin intermediul ferestrei *Accounts* (meniul **Tools** comanda **Accounts**...).

Fereastra conține etichetele: *All, News, Mail, Directory Services.*

Pentru a crea un nou *account* se alege eticheta *Mail*, apoi se apasă butonul **Add** și apoi butonul **Properties**.

Fereastra *Properties* oferă etichetele: *General, Servers, Connection, Security, Advanced.*

Configurarea trimiterii mesajelor cu semnătură

Din meniul **Tools** se alege comanda **Stationery...**; se apasă butonul **Signature...**; se redactează apoi conținutul semnăturii (de obicei informații despre utilizator), acest conținut fiind automat atașat la sfârșitul oricărui mesaj trimis.

Citirea și trimiterea de mesaje și fișiere

Citirea și trimiterea de mesaje și fișiere se realizează prin *Internet Explorer* (folosind funcțiile **Read Mail**, respectiv **New Message**), sau direct prin lansarea în execuție a programului *Outlook Express* care oferă o fereastră cu meniul: **File, Edit, View, Go, Tools, Compose, Help** și butoanele:

- **Compose Message**: compunerea (trimiterea) de mesaje;
- **Reply to Author**: răspunsul la un mesaj primit de la un expeditor (autor);
- **Reply to All**: răspunsul la un mesaj pentru toți cei care l-au primit;
- **Forward Message**: trimiterea mai departe a mesajului;
- **Send and Receive**: trimiterea și primirea de mesaje;
- **Address Book**: lista de adrese *e-mail* utilizate mai frecvent ca destinații.

Meniul **Tools** oferă comanda **Options** prin care utilizatorul poate să seteze diverse opțiuni privind primirea și trimiterea de mesaje, aceste opțiuni fiind disponibile în fereastra *Options* care oferă etichetele *General, Send, Read, Spelling, Security, Dial Up* și *Advanced.*

Pentru compunerea (trimiterea) unui mesaj se acționează sub *Internet Explorer* funcția **Mail | New Message**, sau sub *Outlook Express*, se acționează **Compose Message**, în ambele cazuri apare fereastra pentru compunerea mesajului.

Pentru a trimite un mesaj se completează antetul (*To, Subject*, și după caz *Cc* și *Bcc*), iar apoi se redactează conținutul mesajului. Pentru atașarea (trimiterea) de fișiere la mesajul ce urmează a fi trimis, se utilizează meniul **Insert** care oferă funcțiile **File Attachment** și **Text from File**.

Netscape Communicator

La fel ca și în cazul programului *Internet Explorer*, anterior primei utilizări trebuie să se realizeze diverse setări și să se aleagă din opțiuni.

Fiecare utilizator-client (*guest*) trebuie să-și creeze propriul său *profile*. Această acțiune se realizează prin: **Start | Programs | Netscape Communicator | Utilities | User Profile Manager.**

Se selectează butonul **New** și va apărea o fereastră pentru introducerea numelui și a adresei *e-mail*. După completarea acestor date se apasă butonul **Next>**, ceea ce duce la apariția unei alte ferestre unde trebuie introdus un *login name* și o *adresă IP* sau adresă de domeniu pentru primirea mesajelor.

După apăsarea butonului **Next>** va apărea o nouă fereastră unde trebuie introdusă o adresă IP sau o adresă de domeniu pentru trimiterea mesajelor.

Se apasă din nou butonul **Next>** și va apărea ultima fereastră unde trebuie ales serverul pentru grupurile de știri (de obicei aici nu se introduce nici o informație).

După apăsarea butonului **Finish**, noul profil este creat.

După efectuarea acestor setări, se poate lansa în execuție programul de navigare *Netscape Communicator* care oferă meniul: *File, Edit - Preferences, View, Go, Communicator - Navigator, Messenger, Composer, Help.*

Pentru realizarea unor setări privind preferințele de lucru, se alege **Edit | Preferences** și se optează pentru diverse aspecte (categorii): *Appearance, Navigator, Mail & Newsgroups, Roaming Acces, Composer, Offline, Advanced.*

După configurare, utilizatorul poate să utilizeze serviciile de *e-mail* prin intermediul programului *Netscape Communicator.*

Pe ecran va apărea fereastra pentru alegerea profilului (*Profile Manager*); se alege profilul și apoi se apasă butonul **Start Communicator**.

Pentru a utiliza serviciile *e-mail* se acționează butonul **Inbox** ceea ce va duce la apariția unei ferestre care va cere introducerea parolei pentru contul utilizatorului.

Serviciul *Inbox* gestionează poșta locală (*Local Mail*) utilizând câteva directoare (folders) implicite:

- **Inbox**: mesajele citite temporar de pe server;
- **Unsent Message**: mesajele încă netrimise;
- **Draft**: mesaje în proiect;
- **Templates**: mesaje model;
- **Sent**: mesaje trimise;
- **Trash**: mesaje marcate pentru ștergere

În vederea trimiterii unui mesaj (la care se poate atașa un fișier) se selectează butonul **New Message**.

După redactarea mesajului și eventuala atașare a fișierelor, acesta se trimite folosind butonul **Send**, eventual cu opțiunea de alegere a modului de trimitere (*Plain Text and HTML, Plain Text Only, HTML Only*).

Comentariu

Mesajele citite de pe server în poșta locală a unui utilizator trebuie să fie gestionate de utilizatorul respectiv în scopul evitării ca aceste mesaje să fie citite și de alți utilizatori.

În acest sens, utilizatorul trebuie să-și salveze mesajele și apoi să le șteargă din *Local Mail.*

În cazul sistemului de operare *Windows NT* această măsură de securitate nu mai este necesară.

Transfer de fișiere prin serviciul FTP

Serviciul FTP (*File Transfer Protocol*) realizează transferul de fișiere între mediul *Internet* și utilizator.

Utilizarea serviciului FTP se face în diverse situații (sub *Linux/Unix*, sub *Windows*, sub un program de navigare etc.)

FTP sub Linux/ Unix

Transfer pe server: se execută comanda `ftp` *host* care duce la apariția *prompter*-ului `ftp`; apare *prompter*-ul **`ftp>`**, care va interpreta comenzile introduse de utilizator. Transferul se realizează prin comanda **`put`**:

```
ftp>put <specificator>
```

Transfer de pe server: se realizează prin intermediul comenzii **`get`**:

```
ftp>get <file> [<specificator>]
```

FTP prin browser

Se introduce URL-ul corespunzător și apoi se alege fișierul care trebuie transferat. Utilizarea unui program de navigare este mult mai comodă în cazurile în care transferul de fișiere trebuie să se realizeze de la o locație FTP.

Comentariu

În general, există diverse adrese FTP clasificate pe diverse categorii de domenii cu ajutorul cărora utilizatorii pot să-și transfere diverse produse și informații.

Domnul Marin Vlada este lector la Universitatea București și poate fi contactat la adresa vlada@math.math.unibuc.ro.

Premii ONI 2000

Clasa a IX-a

Premiul I:
Margine Cornel, Moldova
Vernescu Victor, Constanța
Berinde Radu, Bucuresti

Premiul II:
Mihai Cristian, Botoșani
Ciobica Ștefan, Suceava
Cracan Arcadie, Iași
Trufin Valentin, Suceava
Raianu Mihai, Constanța

Premiul III:
Dăneț Tiberiu, București
Matei Claudiu, București
Costan Victor, București
Jucovschi Constantin, Rep. Moldova
Giurgiu Andrei, București
Cîrstea Cristian, Arges
Ardelian Florin, Hunedoara
Pașca Flaviu, Bistrița Năsăud

Clasa a X-a

Premiul I:
Bors Vencel, Bihor
Toth Cristian, Timiș
Dumitran Daniel, București

Premiul II:
Machedon Maximilian, București
Crăciun Ciprian, Hunedoara
Popovici Dan, Arad
Păunescu Liviu, Constanța
Găvănescu Laviniu, Gorj
Hrjoc Bogdan, Alba

Premiul III:
Oghina Andrei, București
Tofan Tiberiu, Galați
Dascălu Vlad, Bacău
Benea Andrei, Vrancea
Vîlceanu Vlad, Arad
Nicolae Bogdan, Sibiu
Radu Vlad, București
Vlaicu Florin, Constanța
Băicoianu Cristian. Prahova

Clasa a XI-a

Premiul I:
Pătrașcu Mihai, Dolj
Ghețu Florin, Vrancea
Alexandrescu Cristian, Botoșani
Crișan Daniel, Vâlcea

Premiul II:
Gheorghe Andrei, București
Patcas Csaba, Bihor
Stan Bogdan, Prahova
Sabău Florin, Caraș Severin
Trifan Ionuț, Buzău
Baltă Mihnea, Constanța
Piloca Bogdan, Timiș
András Csaba, Bihor
Dincă Cristian, Olt

Premiul III:
Petre Răzvan, Constanța
Andreica Mugurel, București
Carabaș Costel, Sibiu
Runcanu Andrei, Gorj
Alexandru Cătălin, București
Diaconescu Radu, Olt
Costescu Andrei, Caraș Severin
Nedelcu Alexandru, Călărași
Tuhuț Adrian, Galați
Barbu Marius, Dolj
Harabagiu Valentin, Prahova

Clasa a XII-a

Premiul I:
Proorocu Angel, București
Leu Ioan Tudor, Vrancea
Ștefan Radu, Brașov
Oprean Daniel, Alba

Premiul II:
Kucszapski Artur, Bihor
Andrei Marius, București
Matei Adrian, Prahova
Stroe Bogdan, București
Prihoi Octav, Sibiu

Premiul III:
Bogdănaș Denis, Rep. Moldova
Ileană Ioana, Alba
Cornea Adrian, București
Mihăilă Cornel, Brașov
Constantin Cătălin, Argeș
Anghel Marian, Ilfov
Dinca Sergiu, Dâmbovița
Fulger Dan, București
Iană Ionel, Olt
Popa Alexandru, București
Popa Claudiu, București
Nicolescu Bogdan, București

CONSTANȚA 3-9 MAI 2000
OLIMPIADA NATIONALA DE INFORMATICA

Așteptăm scrisorile voastre!

POȘTA redacției

Am tot promis că această rubrică va deveni una permanentă în GInfo, dar, din motive de spațiu, a trebuit să renunțăm la ea în ultimele numere. Sperăm că, totuși, la toamnă, dialogul cu cititorii se va concretiza și prin intermediul acestei rubrici. Până atunci, să răspundem acum la câteva scrisori...

Mihai Pătrașcu

Luând la cunoștință cu indignare aprecierile malițioase la adresa sa, subsemnatul a considerat necesar să solicite drept la replică *Gazetei de Informatică*, precizând:

1. sursa pentru problema labirintului, etichetată cu calificativul *GInfo Best Encrypted Solution* nu a fost scrisă pentru a fi înțeleasă de altcineva în afară de autorul însuși; autorul îi invidiază pe cei care au timp suficient pentru a dedica concursului GInfo, dar recunoaște că el însuși dedică majoritatea timpului studiului compilării limbajelor de programare;
2. presupunerea că folosirea denumirilor în limba engleză a făcut sursa dificil de înțeles pare absurdă; algoritmul și micile artificii folosite sunt totuși destul de cunoscute în anumite cercuri de programatori cu experiență; dacă redacția GInfo dorește să dovedească înclinația subsemnatului spre a scrie programe greu de înțeles, îi asigur că există surse mai concludente în acest sens;
3. definirea macroului `TEST` care nu este folosit niciodată nu a fost intenționată; programatorii cu experiență vor recunoaște desigur problema: la un anumit moment dat, sursa a fost rescrisă în proporție de peste 50% din motive de eficiență; neeliminarea macroului în cauză este desigur o scăpare regretabilă, dar total inofensivă; consider însă că problemele implicate de versiuni multiple sunt în general mult mai grave și greu de rezolvat.

Introducerea casetei în care era criticată acea sursă nu a vrut să aducă nici o ofensă autorului, ci să tragă un semnal de alarmă pentru cei care folosesc asemenea practici pentru a demonstra câtă programare știu. Nu sunteți singurul care trimite la redacție surse aproape indescifrabile, dar comentariul a apărut în dreptul acestei surse datorită faptului că era singura care a obținut 100 de puncte la acea problemă și merita să fie publicată. Nimeni nu a presupus că sursa e greu de înțeles din cauza denumirilor în limba engleză, iar redacția GInfo nu are nicidecum intenția de a dovedi anumite înclinații ale vreunei persoane. Referitor la macroul `TEST`*, scăparea este evident doar atât, o scăpare, și nimeni nu pune la îndoială calitățile de programator ale autorului sursei.*

Felicitări pentru medalia de argint de la BOI 2000, pentru clasarea pe locul II la concursul Bursele Agora și succes la faza finală, la CEOI 2000 și IOI 2000.

Diodor Bițan

Felicitări pentru singura revistă de inițiere în programare din România, pentru serialele ("telenovelele") care aprofundează un domeniu, ca și pentru articolele scurte care doar prezintă idei de bază. O singură problemă aș avea. Revista este prea mică. În afară de problemele de olimpiade cu sursele (din care s-ar putea scurta foarte mult, prezentând doar ideea și făcând disponibile sursele complete pe Web), mai rămân câteva pagini. Și dacă mai renunțați și la *Java&Web*, care, deși foarte interesante, nu au mare legătură cu programarea, cu algoritmii... Din punctul meu de vedere, ar fi superb ca revista să aibă măcar 100, 150 de pagini și un CD-ROM, oricât ar costa.

Mulțumim pentru aprecieri și ne bucurăm că GInfo este pe placul dumneavoastră. Revista este într-adevăr prea mică, dar creșterea numărului de pagini ar duce la creșterea costurilor de producție și a prețului revistei care este și așa mult mai mare decât am dori. Și noi ne-am gândit să prezentăm în paginile revistei doar ideile de rezolvare ale problemelor, iar sursele să fie disponibile pe Web, dar acest lucru nu ar fi deocamdată util din simplul motiv că o mare parte a cititorilor au acces limitat la Internet. La Java&Web am putea renunța, dar deocamdată nu am făcut-o deoarece aceasta este o foarte apreciată secțiune a revistei. Într-adevăr acestea nu prea au legătură cu programarea, dar GInfo nu se vrea a fi doar o revistă de inițiere în programare, ci o revistă de inițiere în IT&C. Cât despre CD-ROM, acesta va apărea imediat ce ne vom putea permite. Referitor la faptul că ați fi dispus să cumpărați GInfo indiferent de preț, vă putem spune că un eventual GInfo color cu 100 de pagini și CD-ROM inclus ar costa aproape 100000 lei în condițiile actuale.

COMPUTER PRESS AGORA

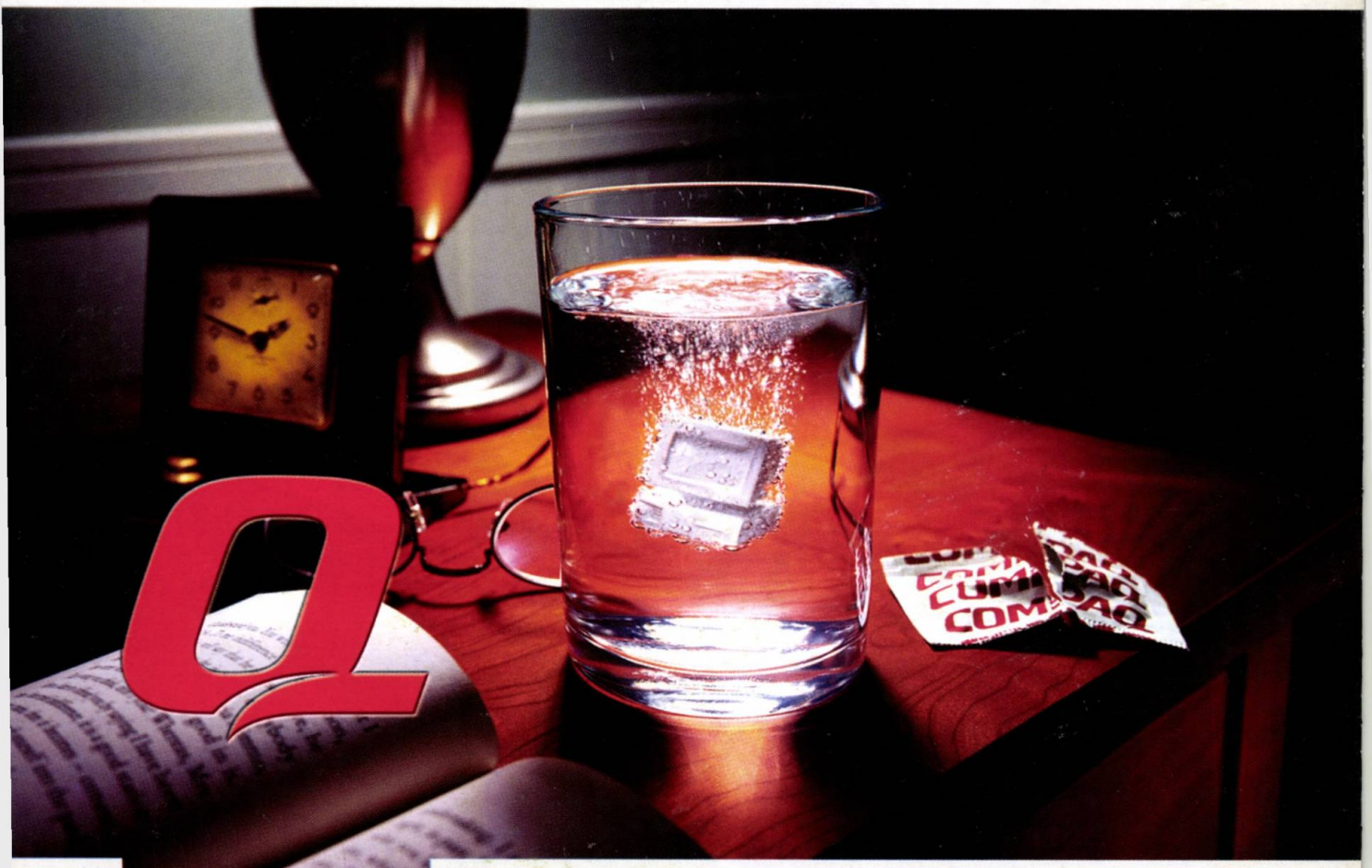
PC REPORT
ECHIPAMENTE · SISTEME · APLICAȚII · REȚELE · INTERNET
Vol. 9, Nr. 1. Ianuarie 2000 LEI 20.000

Din acest număr:

- **Unelte de modelare UML**
- **Introducere în Python**
- Ghid anti-hacker
- Solid Edge 7
- Standarde GNU
- XML prin SAX
- PC Concrete

www.pcreport.ro

Revista
specialiștilor
în tehnologia informației și comunicațiilor



Cure-all?

Dacă vrei să dormi bine noaptea, încearcă Compaq Deskpro. El te eliberează de durerile de cap din timpul zilei, generate de managementul sistemelor de calcul, pentru că este furnizat cu utilitare şi facilităţi deosebite. Dintre acestea, Compaq Insight Manager LC este primul program creat de industria de profil pentru coordonarea unor grupuri de PC-uri printr-o simplă interfaţă bazată pe browser. Cu noile facilităţi te bucuri de o formă fizică perfectă şi mai mult chef de muncă pentru că nu mai pierzi inutil timp pentru depanări şi câştigi ore preţioase de lucru efectiv. Este stimulentul pe care ţi-l oferă valoarea produselor Compaq, este motivul pentru care Deskpro este cel dintâi. Pentru alte idei fortifiante, vizitează site-ul www.compaq.com/deskpro sau contactează-ne la telefon: +40-1-3033100.

intel inside
pentium® III

COMPAQ

prin distribuitorul autorizat

OMNILOGIC BGS

135, Bucureşti-Ploieşti Road,
Bucharest 1, Romania
PO BOX 18-131
Tel: +40-1-303 31 00
Fax: +40-1-303 31 64
www.omnilogic.ro

Intel, Intel Inside şi Pentium sunt mărci înregistrate ale Intel Corporation

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

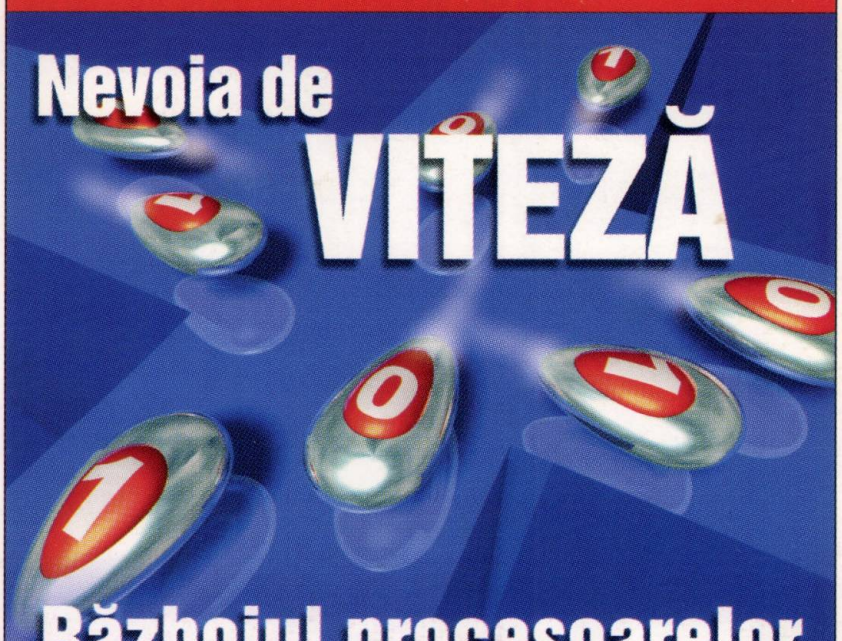
Octombrie 2000
Vol. 10/6

Lei 15.000

din sumarul numărului:

FOCUS

Nevoia de VITEZĂ



Războiul procesoarelor

AMD

intel VS

TEHNO

ACTUAL

CEOI 2000

Olimpiada de Informatică a Europei Centrale

CLUJ NAPOCA - ROMANIA

BABEL

SETI@home

Concursul de PROGRAMARE

CONCURSUL DE PROGRAMARE

AGORA

Cum să pui puterea informației în slujba ta?

Cum să hrănești mai multe gu
deodată? Și cum să le asigu
toate cele necesare fă
întrerupere?! Non stop?!
Simplu - noua generație d
servere Compaq ProLiant ML53
îți sare în ajutor.
Compaq ProLiant ML530 c
procesor Intel® Pentium® I
Xeon™ 866 MHz, cu configurați
dual-procesor este cel mai rapi
server cu arhitectură 2-way di
lume. ProLiant ML530 este gând
pentru exploatarea spațiulu
interior la maxim: până la 4 G
de memorie; 16 bay-uri; 8 slotu
PCI; capacitate maximă d
depozitare a datelor: 436,8 G
Și asta nu e tot: fiabilitatea
stabilitatea sistemului su
garantate prin programe
proprietare Compaq.

intel inside
pentium® III xeon™

Intel, logotipul Intel Inside, Pentium sunt mărci înregistrate, iar Pentium III Xeon este marcă comercială a Intel Corporation.

COMPAQ

NonStop™

scop COMPUTERS

Barbu Văcărescu 162
Tel: 231 65 78
Fax: 231 65 96
www.scop.ro

Vol. 10, Nr. 6 /2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Au mai colaborat:
Diodor Bițan (București)
Militon Frențiu (Cluj)
Doru Popescu Anastasiu (Slatina)
Mihai Stroe (București)

Secretar general de redacție:
Gabriela Bucșa

Tehnoredactare:
Călin Borz
Coperta și grafica:
Andrei Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

editorial.. PAG. 3

Prietenul TĂU

actual.. PAG. 5

La hotel TRANSILVANIA

Olimpiada de Informatică a Europei Centrale

probleme.. PAG. 7

propuse pentru REZOLVARE

Cele 6 probleme de la CEOI 2000

concurs.. PAG. 12

Etapa finală

Regulament

Enunțurile de la prima rundă

focus.. PAG. 22

Principiul INCLUDERII și EXCLUDERII
Doru POPESCU ANASTASIU

Nevoia de VITEZĂ
Diodor BIȚAN

babel.. PAG. 32

SETI@home
Mihai SCORȚARU

tehno.. PAG. 38

Războiul PROCESOARELOR
Mihai SCORȚARU

profil.. PAG. 43

Windows ME

Noul sistem de operare de la Microsoft

begin.. PAG. 44

Conceptul de ALGORITM și importanța lui
Militon FRENȚIU

Admitere la UBB

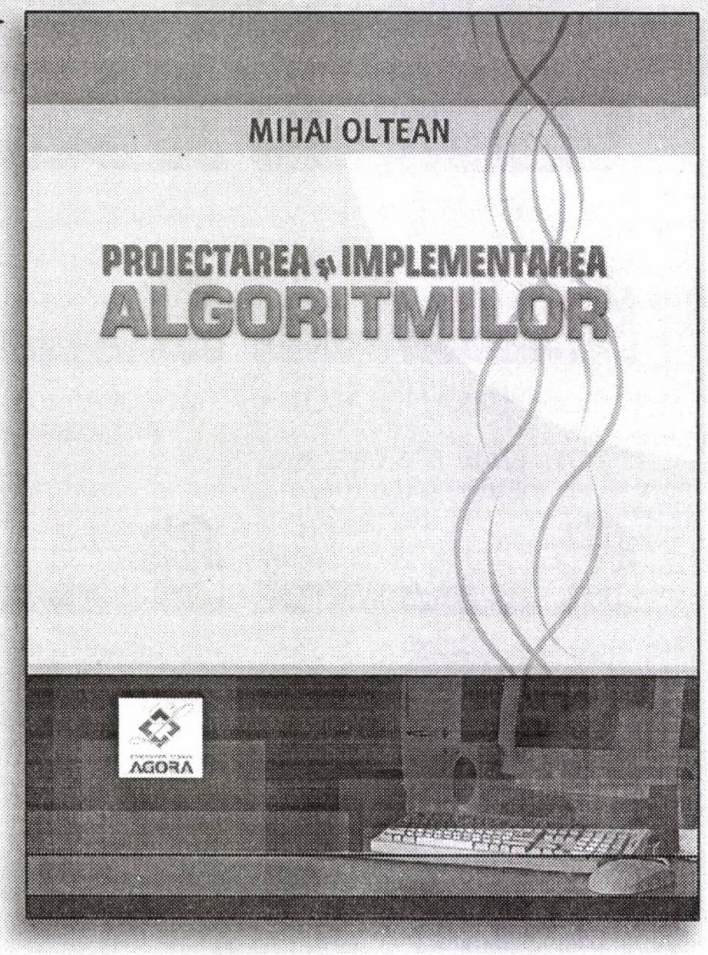
dialog.. PAG. 48

POȘTA redacției

Răspunsuri la scrisori

Așteptăm scrisorile voastre!

COMPUTER LIBRIS AGORA



"Cum aș putea să justific existența acestei cărți? Cum aș putea spre exemplu să justfic cele 100 de pagini despre programarea dinamică, acum când această metodă nu mai figurează în programele școlare? Nu știu. Sau, cum aș putea să justific capitolul despre algoritmi genetici, când pe plan mondial abia de câțiva ani a fost dovedită utilitatea și eficiența lor? Din nou nu știu! Dar rămâne bănuiala: a scrie algoritmi înseamnă a crea. Și nu poți crea fără să gândești."

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Prietenul TĂU

Încă un nou an școlar? De ce "încă unul"? Nu uitați - noi profesorii rămânem mereu corigenți... Ce aș putea să vă urez, ceva nou, prin intermediul paginilor *GInfo*? Să aveți un an școlar cât mai plin de succese? Probabil, această urare este atât de uzată, încât nici nu m-ați auzi... Să aveți zile frumoase și pline de satisfacții? Da, îmi place mai mult, dar tot nu am găsit acel *ceva* care să mă liniștească în sensul că am găsit o urare sinceră și frumoasă care să vă ajute să mergeți înainte, cel puțin cu un mic sfat original din partea *GInfo*.

Dragi cititori ai revistei - elevi, studenți, profesori și părinți! Vă doresc să aveți talentul și voința să descoperiți că *GInfo* vă este prieten. Cum ar putea să fie o revistă totodată și prieten? În primul rând conține informații de care aveți nevoie. Apoi, redacția are porțile deschise și așteaptă, nu doar sugestiile voastre, dar și materiale pregătite pentru a fi publicate în revistă. Vă ajută să vă antrenați în de-ale programării..., iar pe cei mai buni îi sprijină cu burse.

Recunosc - mi-ar fi plăcut mai mult să scriu *cel mai bun prieten*, dar consider că acest atribut trebuie păstrat pentru *o persoană*, adolescent sau adult, fată sau băiat, depinde de fiecare.

Mă învârt în jurul urării și descopăr acest subiect minunat cu care aș putea continua: *prietenia*... Presupunând că majoritatea cititorilor, sunteți mai tineri decât mine, permiteți-mi să vă împărtășesc ceva din experiența mea. Cele mai "grozave" prietenii le-am legat în perioada adolescenței. Care este legătura cu voi? Aș vrea să vă transmit ceva ce se formulează greu, dar merită să încerc: îngrijiți zi de zi prieteniile voastre. Încercați să descoperiți bucuria pe care o veți simți, "știind" - nu doar sperând - că puteți avea încredere în prietenul vostru. Cum îl descoperiți? Este greu să vă răspund, dar mă voi hazarda din nou: un prieten adevărat te ascultă cu răbdare și la rândul lui îți împărtășește grijile; un prieten adevărat nu îți va spune "ai fost grozav", când nu ai fost și nu se va supăra când, la rândul tău, vei fi la fel de sincer; un prieten va fi mereu alături (va ști mereu unde și când) dacă vei avea nevoie de el. Va simți respectul tău și te va respecta. Va face echipă cu tine, nu va dori să îți dea lecții în fiecare moment, în schimb "se va sfătui cu tine".

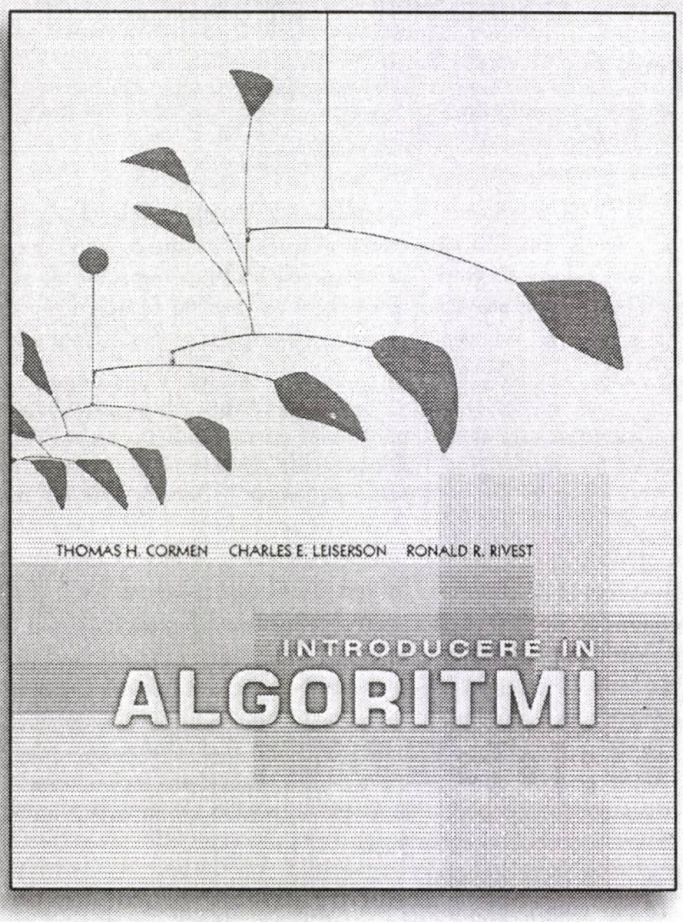
Îi văd pe cititorii adulți, strâmbându-se: *povești*... Da, pe de altă parte există proverbul românesc: "*Doamne, Dumnezeule ferește-mă de prieteni, că de dușmani mă feresc eu singur.*" Oare de câte ori în viață ne-am adus aminte de acest proverb trist? Depinde de noroc, de personalitate și de multe altele. Chiar dacă un fost prieten nu îți devine dușman, te poate necăji și altfel. De exemplu, poate "coborî de pe piedestal". Ce înseamnă asta? Vă minte într-o zi, absolut inexplicabil. Vă roagă apoi să faceți și voi ceva similar. Vă "ajută" să coborâți și voi de pe un alt piedestal din ochii altui prieten. Și atunci ce se va întâmpla? Cum arată o lume fără prieteni, fără prietenie? Greu de descris. Imaginați-vă o lume în care copiii nu pot să se joace, pentru că nu există jucării... E cam același lucru.

"Anii de școală sunt cei mai frumoși ani din viață" vi se spune deseori, dar nu credeți, nu-i așa? "Este groaznic, trebuie să mergeți zilnic la școală", acolo vă "chinuie fel de fel de profesori" care bineînțeles au fost inventați cu un singur scop: "să vă facă capu' mare". Dar tot acolo, la școală vă întâlniți cu colegii de clasă, sau din alte clase, printre care - undeva - se ascunde un *El* sau o *Ea*, *prietenul* vostru. Căutați-l, vă așteaptă cu siguranță.

Revenind la urarea mea, vă doresc un an școlar pe care să-l țineți minte exact din cauza că ați legat prietenii grozave, trainice, un an care nu va dispărea pur și simplu în trecut. Iar oamenilor mari le doresc să-și întâlnească prietenii cât mai des, să se simtă bine cu ei și să uite proverbul amintit.

Clara Ionescu

COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

La hotel TRANSILVANIA

Mihai Scorțaru

Cea de-a șaptea ediție a Olimpidei de Informatică a Europei Centrale a luat sfârșit. Spre surprinderea multora, totul s-a desfășurat în cele mai bune condiții, domnul Ivo Separovic, team lider-ul echipei Croației a declarat că acesta a fost cel mai bine organizat concurs de informatică la care a participat.

CEOI 2000 a fost organizat de *Ministerul Educației Naționale* prin *Inspectoratul Școlar Județean Cluj* în colaborare cu editura *Computer Press Agora*, editorul *Gazetei de Informatică*.

Concursul s-a desfășurat la *Hotelul Transilvania* situat în mijlocul orașului. Între 24 și 31 August 2000 acest hotel a fost centrul informaticii central-europene.

După șase ani, CEOI s-a întors la *Cluj*, orașul care a găzduit și prima ediție.

Majoritatea participanților au sosit pe data de 24 August. A doua zi a avut loc festivitatea de deschidere, în *Aula Magna* a *Universității Babeș-Bolyai*. Prima zi de concurs a fost 26 August; a urmat o zi de pauză și apoi a doua zi de concurs. În ziua de 29 August a fost organizată o excursie la stațiunea *Beliș-Fântânele*. Festivitatea de premiere a avut loc pe data de 30 August în sala de spectacol a *Operei de Stat*, iar pe 31 August participanții au părăsit orașul *Cluj*.

La acest concurs au participat cele opt țări membre CEOI (*Cehia*, *Croația*, *Germania*, *Polonia*, *România*, *Slovacia*, *Slovenia* și *Ungaria*) și trei țări invitate (*Moldova*, *Olanda* și *Statele Unite ale Americii*). Fiind țara organizatoare *România* a avut trei echipe în concurs. *Germania* a participat pentru prima dată ca membru *CEOI*, *Olanda* și *Moldova* au fost invitate pentru a doua oară, iar *Statele Unite ale Americii* pentru a treia oară.

Acest concurs a fost folosit și pentru a selecționa echipa *României* care va participa la *Olimpiada Internațională de Informatică* care se desfășoară anul acesta la *Beijing*, în *Republica Populară Chineză*.

Cei patru reprezentanți ai *României* la IOI 2K sunt:

Daniel Dumitran, București
Mihai Pătrașcu, Craiova
Angel Proorocu, București
Radu Ștefan, Brașov

Ei vor fi însoțiți de dl. *prof. dr. Horia Georgescu* (coordonator științific al *GInfo*) și d-na *lect. univ. Clara Ionescu* (redactor-șef al *GInfo*), cei doi inițiatori ai CEOI-ului.

Peste o sută de persoane au fost implicate în organizarea acestui eveniment. Concursul a fost sponsorizat de peste 25 de companii din industria IT&C românească.

Calculatoarele folosite în timpul concursului au fost oferite de firmele *Brinel* și *IBM*. Au fost cele mai performante calculatoare folosite la un concurs de programare din lume, ele

Opinii...

Ioana Dimitriu, Reporter CEOI Newsletter

CEOI 2000 a însemnat mai mult decât un concurs... a însemnat o ECHIPĂ de oameni minunați, de la organizatori, până la concurenți și invitați. M-au uimit organizarea impecabilă, eficiența cu care toată echipa lucra, puterea de care toți au dat dovadă, dar mai ales faptul că și la 2, 3 noaptea, în ciuda oboselii, cei mai mulți își păstrau simțul umorului.

Octavian Costache, organizator CEOIbyNET

Succesul unui proiect se măsoară prin părerile celor cărora li se adresează. Inevitabil, nu toate părerile despre CEOIbyNET au fost excelente, dar majoritatea covârșitoare a felicitărilor pe care le-am primit din partea participanților m-au făcut să cred că am reușit. De aceea aș dori să mulțumesc și organizatorilor fără de care această idee nu ar fi fost pusă în practică.

fiind dotate cu procesoare *Intel Celeron* cu frecvența de 600 MHz. Ca un termen de comparație, putem să vă spunem că la IOI, în *China*, concurenții folosesc calculatoare dotate cu procesoare *Intel Celeron* cu frecvența de 300 MHz.

Sistemul de operare folosit a fost MS DOS 6.22, iar mediile de programare puse la dispoziție au fost *Borland Pascal 7.0* și *Borland C++ 3.1*.

Accesul *Internet* a fost asigurat de *RoEduNet* și *Xnet*, toți participanții având posibilitatea de a lua legătura cu restul lumii. Premiile au fost substanțiale, fiind oferite de principalii sponsori ai concursului.

Lista completă a medaliaților la această ediție a *Olimpiadei de Informatică a Europei Centrale* este prezentată la pagina 11.

Comitetul științific și comitetul de evaluare de la CEOI 2K au fost formate din studenți și profesori. Sistemul de evaluare a fost creat de domnul ***Felméri Péter***, programator la firma *Simbolic* din *Cluj*. A fost același sistem pe care comisia l-a testat, în mai 2000, cu ocazia *Olimpiadei Naționale de Informatică* de la *Constanța* unde, din nefericire, nu au fost create condiții optime de funcționare. Acest sistem de evaluare, în forma folosită la CEOI, a fost apreciat de domnii ***Tom Verhoeff*** din *Olanda* și ***Gyula Horvath*** din *Ungaria* (membri în *Comisia Științifică de Supervizare* a IOI) ca fiind mai performant decât cel care se va folosi în *China*, deoarece evaluarea are loc pe stația de lucru a concurentului, câștigându-se timp.

La festivitatea de premiere și la cina festivă au fost invitate formațiile clujene *Semnal M* și *Acces*.

Cu ocazia acestui concurs a apărut o ediție specială a revistei noastre, numită *GInfo Special Edition - CEOI 2000*.

Majoritatea participanților la *Olimpiada de Informatică a Europei Centrale* se vor revedea în *China* la IOI. Este cunoscut faptul că gradul de dificultate al problemelor de la CEOI este mai mare decât cel al problemelor de la IOI. Aceasta se datorează nivelului ridicat de pregătire al elevilor din acestă zonă.

CEOI este cel mai puternic concurs de programare pentru elevi din întreaga lume. Dovezi în acest sens sunt numeroase, cele mai concludente fiind dorința a numeroase țări de a participa la acest concurs (anul acesta au avut noroc *Statele Unite ale Americii* și *Olanda*, chiar dacă și elevi din *Indonezia* sau *China* ar fi dorit să participe) și numărul mare de medalii obținut de participanții din țările central-europene la ultimele ediții ale *Olimpiadei Internaționale de Informatică*.

Continuând tradiția ultimilor ani, cei care nu au fost prezenți la concursul oficial de la *Cluj-Napoca* au putut participa la concurs neoficial desfășurat prin intermediul rețelei *Internet*. Participanții la acest concurs neoficial (*CEOIbyNet*) au încercat să rezolve aceleași probleme ca și cei 51 care luptau pentru medalii. *CEOIbyNet* a fost organizat de *dl. prof. Emil Onea* și de *Octavian Costache*. Și acest concurs s-a desfășurat în cele mai bune condiții, numeroși participanți din toate colțurile lumii având cuvinte de laudă la adresa *CEOIbyNet*.

Conform tradiției de fiecare echipă trebuie să "aibă grijă" o ghidă sau un ghid. La CEOI 2000 acest serviciu a fost asigurat de eleve și elevi, majoritatea de la *Liceul de Informatică* din *Cluj*. Au fost apreciați nu doar pentru buna cunoaștere a limbii engleze, ci și pentru ospitalitatea de care au dat dovadă.

Chiar dacă organizatorii *Olimpiadei de Informatică a Europei Centrale* au trecut prin multe emoții, totul s-a desfășurat conform planului, micile întârzieri (suntem totuși în *România*!) neafectând programul participanților. În cele din urmă toți cei implicați au fost nevoiți să recunoască faptul că cea de-a șaptea ediție CEOI a fost un succes.

Cea de-a opta ediție a *Olimpidei de Informatică a Europei Centrale* va avea loc anul viitor în orașul *Zalaegerszeg* din *Ungaria*.

Mihai Scorțaru a fost membru în Comitetul de Evaluare al celei de-a șaptea ediții a Olimpiadei de Informatică a Europei Centrale. Poate fi contactat prin e-mail la adresa skortzy@email.ro.

Opinii...

Bogdan Dumitru, membru în Comitetul de Evaluare

Anul trecut am participat la CEOI '99. Atunci concuram pentru echipa României. Anul acesta, când CEOI-ul s-a organizat în România, am avut șansa extraordinară de a participa din nou, de data aceasta de "cealaltă parte a baricadei", adică de partea comisiei, comisie care, de data aceasta, a fost formată în mare parte din studenți. A fost creată o atmosferă de lucru pe care eu personal nu am mai întâlnit-o. Fiecare om avea sarcini precise de executat, pe care reușea să le ducă la bun sfârșit în timpul cerut, lipsind în același timp stresul sau încordarea.

Fără falsă modestie, cred că pot să afirm că partea de evaluare a surselor concurenților (principala sarcină a Comitetului de Evaluare) a fost la un înalt nivel.

CEOI 2000

Vă prezentăm acum enunțurile celor șase probleme cu care s-au confruntat cei 51 de participanți la cea de-a șaptea Olimpiadă de Informatică a Europei Centrale. Problemele au fost propuse de Comitetul Științific al CEOI format din Iuliu Vasilescu, Dana Lica, Cătălin Frâncu, Mihai Stroe și Radu Lupșa.

P060001: Planeta X

Toți locuitorii planetei *X* își construiesc casele în formă triunghiulară. Pentru a economisi timp și efort, ei au o metodă de construcție specială. Totul începe cu un perete drept. Apoi, pentru a construi orice casă nouă, adaugă doar doi pereți noi la un perete deja existent, rezultând astfel o casă de formă triunghiulară. Bineînțeles, cei doi pereți nou adăugați pot fi de asemenea utilizați ca perete de pornire pentru noi case. Câteodată, utilizând acest model de construcție, apar situații în care unele case sunt complet incluse în altele (ca și în figură). Totuși, aceasta nu este o problemă, deoarece copiii pot locui în casele incluse.

Pentru a-și ilumina casele, locuitorii planetei X montează câte un bec în fiecare colț al construcției finale (un bec dintr-un colț fiind comun tuturor caselor care au acel colț). În fiecare colț există un comutator, prin acționarea căruia se schimbă starea (aprins/stins) a becului din colțul respectiv, precum și a becurilor din colțurile conectate cu acesta. Două colțuri sunt considerate conectate dacă sunt plasate la extremitățile aceluiași perete.

Cerință

Scrieți un program care determină o secvență de acționări ale comutatoarelor, astfel încât, pornind de la o configurație inițială dată, în final să fie aprinse toate becurile.

Intrare

Numele fișierului: `X.IN`

Linia 1: `N`

- un număr întreg, reprezentând numărul de colțuri ale construcției, numerotate de la 1 la `N`;

Liniile `2..2XN-2`: `I J`

- două numere întregi, separate printr-un spațiu, reprezentând un perete între colțurile `I` și `J`;

Linia `2XN-1`: S_1 S_2 ... S_N

- `N` numere întregi (separate prin spații) ale căror valori `0` sau `1` corespund stărilor celor `N` becuri;
- `0` semnifică stins; `1` semnifică aprins.

Se garantează faptul că datele de intrare reprezintă o construcție care poate fi obținută în modul specificat.

Ieșire

Numele fișierului: `X.OUT`

Linia 1: L_1 L_2 ... L_K

- `K` numere întregi, separate prin spații, reprezentând lista etichetelor comutatoarelor acționate.
- Dacă există mai multe soluții, se cere numai una.
- Dacă nu există soluție, veți afișa în fișierul de ieșire o singură linie care conține numărul `0`.

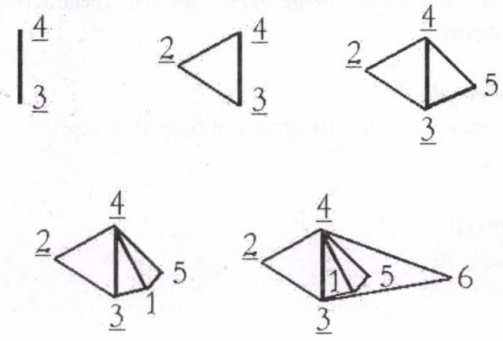
Restricții

- $3 \le N \le 1000$

Exemplu

X.IN	X.OUT
6	1 6
1 3	
1 4	
1 5	
2 3	
2 4	
3 4	
3 6	
4 5	
4 6	
0 1 1 1 0 0	

În figura de mai jos puteți urmări pașii posibili pentru a obține construcția din fișierul exemplu; etichetele becurilor care sunt inițial aprinse sunt subliniate.



P060002: Străzi

Ministerul Transporturilor din *România* decide să îmbunătățească drumurile românești.

Fiecare drum este bidirecțional și conectează direct două orașe.

Între două orașe nu poate exista mai mult de un drum. Rețeaua de drumuri existentă asigură direct sau indirect legătura între oricare două orașe din *România*.

Totuși, pentru îmbunătățirea drumurilor se închide, pe rând, fiecare drum și se execută reparațiile necesare.

Pe durata acestor operații este necesar să se păstreze conectivitatea rețelei de drumuri.

Pentru aceasta, *Ministerul Transporturilor* decide să se construiască noi drumuri, astfel încât, în orice moment, indiferent care drum este închis (în orice moment fiind închis un singur drum), traficul între oricare două orașe să fie în continuare posibil.

Bineînțeles, numărul drumurilor noi adăugate trebuie să fie minim.

În plus, nici un drum nou nu poate lega direct două orașe între care există deja o legătură directă.

Cerință

Scrieți un program care determină numărul minim de drumuri suplimentare care trebuie construite și perechile de orașe legate de acestea.

Dacă există mai multe soluții optime, se cere numai una.

Intrare

Numele fișierului: `ROADS.IN`

Linia `1`: `N M`

- Două numere întregi, separate printr-un spațiu, reprezentând numărul de orașe și respectiv numărul de drumuri. Orașele sunt etichetate de la `1` la `N`.

Liniile `2..M+1`: T_1 T_2

- Două numere întregi, separate printr-un spațiu, reprezentând două orașe legate printr-un drum.

Ieșire

Numele fișierului: `ROADS.OUT`

Linia `1`: K

- Un număr întreg, reprezentând numărul minim de drumuri suplimentare.

Liniile `2..K+1`: C_1 C_2

- Două numere întregi, separate printr-un spațiu, reprezentând o pereche de orașe care vor fi legate printr-un drum.

Observație

- Ordinea perechilor de orașe din fișierul de ieșire nu contează.

Restricții

- $3 \le N \le 2500$
- $2 \le M \le 20000$

Exemplu

ROADS.IN	ROADS.OUT
4 3	2
1 2	1 4
2 3	1 3
2 4	

P060003: Omida

Definiție: O "omidă" este un arbore cu următoarea proprietate: *există un lanț central astfel încât fiecare nod al arborelui fie aparține lanțului central fie este adiacent lanțului.* Figurile de mai jos ilustrează două "omizi", nodurile gri indicând lanțul.

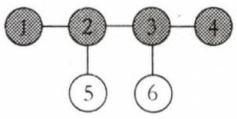


Figura 1

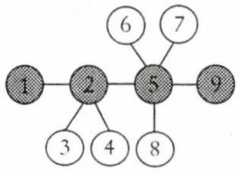


Figura 2

Pot exista mai multe lanțuri. De exemplu, un alt lanț pentru "omida" din figura 2 este 3-2-5-9.

Cerințe

Scrieți un program care, fiind dată o "omidă" cu `N` noduri, va eticheta nodurile cu numere de la `1` la `N` ce satisfac condițiile:

- Fiecare etichetă de la `1` la `N` este folosită o singură dată;
- Nu vor exista două muchii cu același modul al diferenței dintre etichetele nodurilor adiacente.

Mai jos este ilustrat un mod de etichetare a "omidei" din figura 2. Modulele diferențelor sunt indicate pe muchiile respective.

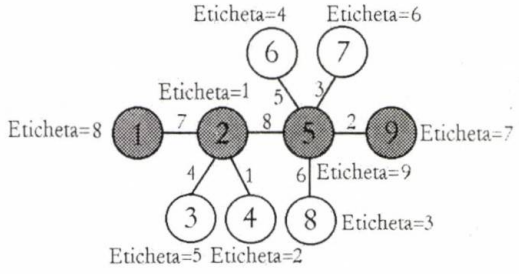


Figura 3

Intrare

Numele fișierului: `CP.IN`

Linia `1`: `N`

- Un întreg care indică numărul de noduri;

Liniile 2..N: X Y
- Două numere întregi, separate printr-un spațiu, reprezentând două noduri adiacente conectate cu o muchie.

Datele de intrare sunt corecte și arborele dat este întotdeauna o "omidă."

Ieșire
Numele fișierului: CP.OUT
Linia 1: L_1 L_2 ... L_N
- N numere întregi, separate prin spații, unde L_i reprezintă eticheta asociată nodului i.

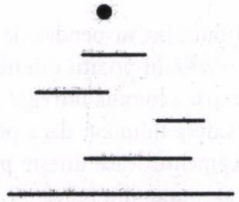
În cazul când există mai multe soluții, se va prezenta numai una. Dacă nu este posibilă nici o etichetare, fișierul de ieșire va conține pe unica linie cuvântul IMPOSSIBLE.

Restricții
- $2 \le N \le 10000$

Exemplu
Fișierul de intrare (**CP.IN**) descrie "omida" din figura 2, iar fișierul de ieșire (**CP.OUT**) descrie etichetarea din figura 3.

CP.IN	CP.OUT (o soluție posibilă)
9	8 1 5 2 9 4 6 3 7
1 2	
6 5	
5 7	
4 2	
2 3	
8 5	
2 5	
5 9	

P060004: Cădere
Considerăm un joc bazat pe un dispozitiv ca în imaginea de mai jos:



Dispozitivul constă dintr-o mulțime de platforme orizontale de diferite lungimi, plasate la diferite înălțimi. Platforma cea mai joasă este podeaua (este plasată la înălțimea 0 și are lungime infinită).

Dintr-o poziție specificată, este lăsată să cadă o minge, la momentul 0. Mingea cade cu viteză constantă de 1 metru pe secundă. Când mingea ajunge pe o platformă, începe să se rostogolească spre unul din capetele acesteia, la alegerea jucătorului, cu aceeași viteză de 1 metru pe secundă. Când mingea ajunge la capătul platformei, își continuă căderea liberă, pe verticală. Mingea nu are voie să cadă liber mai mult de MAX metri odată (între două platforme).

Cerință
Scrieți un program care determină un mod de rostogolire a mingii pe platforme astfel încât să ajungă la podea cât mai repede posibil.

Intrare
Numele fișierului: FALL.IN
Linia 1: N X Y MAX
- Patru numere întregi, separate prin câte un spațiu: numărul de platforme (excluzând podeaua), poziția de plecare a mingii (coordonatele pe orizontală și verticală), și distanța maximă pe care are voie să cadă; platformele sunt numerotate de la 1 la N.

Liniile 2..N+1: $X1_i$ $X2_i$ H_i
- Trei numere întregi, separate prin spații; platforma i este situată la înălțimea H_i, între coordonatele orizontale $X1_i$ și $X2_i$ inclusiv ($X1_i < X2_i$, i=1..N).

Observații
- Diametrul mingii și grosimea platformelor se ignoră. Dacă mingea cade exact pe marginea unei platforme, se consideră că a căzut pe platformă.
- Oricare două platforme nu au puncte comune.
- Pentru datele de test există întotdeauna soluție.
- Toate dimensiunile sunt exprimate în metri.

Ieșire
Numele fișierului: FALL.OUT
Linia 1: TIME
- Un număr întreg, reprezentând timpul la care mingea atinge podeaua, conform soluției voastre.

Celelalte linii, până la sfârșitul fișierului: P T D
- Trei numere întregi, separate prin spații. Mingea atinge platforma P la momentul T și se îndeaptă în direcția D (0 pentru stânga și 1 pentru dreapta).
- Impactul cu podeaua nu trebuie să apară în aceste linii.
- Impactele cu platformele se precizează astfel încât valorile succesive ale lui T să fie în ordine crescătoare.

Observație
- Dacă există mai multe soluții posibile, se cere numai una.

Restricții
- $1 \le N \le 1000$
- $-20000 \le X1_i, X2_i \le 20000$ (i=1..N)
- $0 < H_i < Y \le 20000$

Exemplu
FALL.IN	FALL.OUT
3 8 17 20	23
0 10 8	2 4 1
0 10 13	1 11 1
4 14 3	3 16 1

P060005: Bețișoare

Se consideră următorul joc între doi jucători: pe o tablă sunt plasate N rânduri de bețișoare, pe rândul i fiind plasate S_i bețișoare ($1 \le i \le N$). Bețișoarele de pe fiecare rând i sunt etichetate cu numere consecutive de la 1 la S_i. Jucătorii fac câte o mutare, alternativ. Fiecare mutare constă din eliminarea a unul, două sau trei bețișoare de pe același rând. Aceste bețișoare trebuie să fie etichetate cu numere consecutive, adică să fie așezate unul după altul pe rândul respectiv.

De exemplu, dacă există un rând cu zece bețișoare și primul jucător elimină bețișoarele cu etichetele 4, 5, 6, rămân numai bețișoarele 1, 2, 3, 7, 8, 9 și 10. Al doilea jucător, la rândul său, are posibilitatea de a elimina bețișoarele 1, 2, 3 dar nu și 3, 7, 8 deoarece acestea nu sunt etichetate secvențial (bineînțeles, există și alte mutări valide).

Jucătorul care execută ultima mutare (adică elimină ultimul bețișor de pe tablă) este câștigătorul jocului.

Cerință

Scrieți un program care să implementeze o strategie de câștig împotriva calculatorului.

Intrare

Numele fișierului: **STICKS.IN**

Linia 1: N

- Un întreg, reprezentând numărul de rânduri cu bețișoare ($1 \le N \le 10$).

Linia 2: S_1 S_2 ... S_N

- N numere întregi, separate prin spații, reprezentând numărul de bețișoare de pe fiecare rând ($1 \le S_i \le 10$).

Linia 3: X

- Singurele valori posibile pentru X sunt 0 și 1. Dacă X=0, programul vostru va face prima mutare; altfel, calculatorul va muta primul.

Observație

În datele de test, se garantează existența unei strategii de câștig pentru programul vostru.

Programul vostru va juca împotriva calculatorului. Interfața dintre programul vostru și calculator se realizează cu ajutorul următoarelor elemente soft:

- *Pascal*: unit-ul **STICKS**
- *C/C++*: fișierul header **STICKS.H**

Aceastea conțin următoarele subprograme:

```
procedure putMove(nr,label1,label2:Integer);
                                        (Pascal)
void putMove(int nr,int label1,int label2);
                                        (C/C++)
```

Aceste subprograme se apelează pentru a anunța o mutare a programului vostru, cu semnificația "elimină de pe rândul nr bețișoarele etichetate cu numere de la label1 la label2 inclusiv" (label1≤label2)

```
procedure getMove(var nr,label1,
                  label2:Integer); (Pascal)
void getMove(int *nr,int *label1,int *label2);
                                        (C/C++)
```

Aceste subprograme se apelează pentru a obține o mutare a adversarului, în formatul de mai sus;

Programatorii în C/C++ trebuie să transmită pointeri către variabilele în care doresc să obțineți datele.

Programul trebuie să apeleze alternativ cele două subprograme până când nu mai rămâne nici un bețișor pe tablă. Modulul de program al comisiei va termina jocul dacă programul concurentului va executa o mutare invalidă.

Exemplu

STICKS.IN

```
2
1 3
1
```

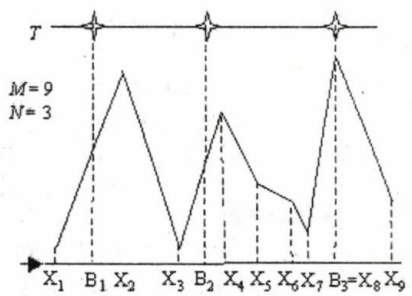
Secvență de apeluri posibilă în Pascal:

```
getMove(nr,k1,k2); → nr=2, k1=2, k2=2
putMove(1,1,1);
getMove(nr,k1,k2); → nr=2, k1=1, k2=1
putMove(2,3,3);
```

◊◊◊ programul cocurentului câștigă ◊◊◊

P060006: Relief luminat

Considerăm un relief compus din segmente de dreaptă unite.



Deasupra reliefului sunt suspendate la aceeași înălțime N becuri. Becurile se află în poziții diferite pe orizontală. Becurile servesc pentru a lumina întregul relief. Un punct de pe relief se consideră iluminat dacă poate "vedea" un bec direct, adică segmentul care unește punctul cu becul nu contine nici un alt punct din relief.

Cerință

Scrieți un program care determină numărul minim de becuri care trebuie aprinse pentru a ilumina întreg relieful.

Intrare

Numele fișierului de intrare: **LIGHT.IN**

Linia 1: M

- Un întreg reprezentând numărul de specificații a înălțimilor reliefului, inclusiv primul și ultimul punct.

Liniile **2..M+1:** X_i H_i

- Două numere întregi separate prin spații: înălțimea H_i a reliefului la poziția orizontală X_i, $1 \le i \le M$; pentru $1 \le i \le M-1$ are loc $X_{i+1} > X_i$; orice două puncte de relief specificate consecutiv identifică un segment de dreaptă al reliefului.

Linia **M+2:** N T

- Două numere întregi separate printr-un spațiu: numărul de becuri și înălțimea lor deasupra reliefului (altitudinea). Becurile sunt numerotate de la 1 la N.

Linia **M+3:** B_1 B_2 ... B_N

- N numere întregi separate prin spații: coordonatele orizontale ale becurilor, $B_{i+1} > B_i$, $1 \le i \le N-1$;

Ieșire

Numele fișierului de ieșire: **LIGHT.OUT**

Linia **1:** K

- Un întreg: numărul minim de becuri care trebuie aprinse.

Linia **2:** L_1 L_2 ... L_K

- K numere întregi, separate prin spații: numerele becurilor care trebuie aprinse, specificate în ordine crescătoare a coordonatelor orizontale.

Restricții

- $1 \le M \le 200$
- $1 \le N \le 200$
- $1 \le X_i \le 10000$ pentru $1 \le i \le M$
- $1 \le T \le 10000$
- $1 \le H_i \le 10000$ pentru $1 \le i \le M$
- $T > H_i$ pentru orice $1 \le i \le M$
- $X_1 \le B_1$ și $B_N \le X_M$
- Pentru testele date vor exista întotdeauna soluții. În cazul în care există mai multe soluții, se va prezenta numai una.

Exemplu

LIGHT.IN	LIGHT.OUT
6	2
1 1	1 4
3 3	
4 1	
7 1	
8 3	
11 1	
4 5	
1 5 6 10	

Medalii...

Aur:

Raid Barton, SUA
Radu Ștefan, România
Tomasz Czajka, Polonia

Argint:

Jan Oravec, Slovacia
Marian Dvorsky, Slovacia
Mihai Pătrașcu, România
Denis Bogdănaș, Moldova
Florin Ghețu, România
Ioana Ileană, România
Tomasz Malesinski, Polonia
Angel Proorocu, România
Peter Pallos, Ungaria

Bronz:

Jozef Tvarozek, Slovacia
John Danaher, SUA
Daniel Dumitran, România
Daniel Jasper, Germania
Dominic Battre, Germania
Arcadie Crăcan, România
Grzegorz Stelmaszek, Polonia
Percy Liang, SUA
Gabor Gyebnar, Ungaria
Marius Andrei, România
Mario Zivic, Croația
Michael Siepmann, Germania
Krzysztof Onak, Polonia

Etapa FINALĂ

Prima ediție a concursului de programare organizat de Gazeta de Informatică s-a încheiat. Primii zece în clasamentul după cele șase runde desfășurate prin e-mail au participat la faza finală care a avut loc pe data de 17 iunie 2000 la Liceul de Informatică din Cluj.

Cei zece participanți calificați la faza finală au fost:

Ciprian Baicu, Drobeta Turnu Severin
Cristian Cadar, București
Adrian Cârcu, Bistrița
Andrei David, Râmnicu Vâlcea
Daniel Dumitran, București
Ioana Ileană, Alba Iulia
Csaba Patcas, Oradea
Mihai Pătrașcu, Craiova
Mihai Stroe, București
Vlad Vâlceanu, Arad

Singurul care nu a putut să se prezinte la faza finală a fost *Cristian Cadar*, așadar am avut doar nouă participanți.

Până la urmă bursa de 50$/lună oferită de *Computer Press Agora* a fost câștigată de *Mihai Stroe*. El a fost singurul care a reușit să obțină punctajul maxim la toate cele trei probleme propuse spre rezolvare. Pe locul al doilea s-a clasat *Mihai Pătrașcu* care a câștigat o bursă de 30$/lună. Toți ceilalți participanți au primit premii oferite de editurile *Computer Press Agora* și *Computer Libris Agora*.

În continuare vă prezentăm enunțurile celor trei probleme împreună cu rezolvările propuse de câștigătorul concursului.

CR7P1: Sens unic

Într-un oraș primarul dorește să introducă sensuri unice pe cât mai multe străzi. Există o singură restricție, și anume din fiecare punct al orașului trebuie să se poată ajunge în orice alt punct, respectând sensurile de circulație.

Planul orașului este dat sub forma unui graf neorientat, conex, vârfurile reprezentând intersecțiile, iar muchiile reprezentând străzile.

Cerință

Ajutați primarul să stabilească străzile cu circulație în sens unic, precum și sensul de circulație. Rezultatul va fi un graf orientat având în locul fiecărei muchii din graful inițial un arc sau eventual o pereche de arce în sensuri opuse.

Date de intrare

Datele de intrare se vor citi din fișierul `STRAZI.IN` având următorul format:

- Pe prima linie se află două numere întregi, N și M, reprezentând numărul de vârfuri, respectiv numărul de muchii; cele două numere sunt separate printr-un spațiu;
- Pe fiecare din următoarele M linii se găsesc câte două numere, reprezentând extremitățile uneia dintre cele M muchii.

Date de ieșire

Rezultatul se va scrie în fișierul `STRAZI.OUT` astfel: pe fiecare linie se va scrie câte o pereche de numere, reprezentând vârful inițial, respectiv cel final, al câte unui arc din graful rezultat.

Restricții

- $N \leq 500$
- $M \leq 3000$
- fiecare muchie apare o singură dată
- vârfurile sunt numerotate de la 1 la N
- în graful dat nu există bucle
- ordinea listării arcelor nu are importanță
- dacă există mai multe soluții optime, se va da doar una singură

Exemplu

`STRAZI.IN`	`STRAZI.OUT`
8 10	1 2
1 2	2 1
2 3	2 3
3 5	3 2
3 6	3 4
3 4	4 5
5 4	4 6
6 4	4 7
4 7	7 8
4 8	5 3
8 7	6 3
	8 4

Soluție

Realizăm o parcurgere *DF* a grafului; pentru fiecare nod calculăm nivelul său. Muchiile arborelui *DF* vor deveni arce de la părinte spre fiu. Pentru fiecare nod, calculăm înălțimea minimă la care putem ajunge din noduri din subarborii ai căror rădăcini sunt fiii nodului respectiv. Dacă nu putem ajunge mai sus (pe un nivel mai mic decât al nodului curent), vom duce un arc de la nodul respectiv în sus, și anume arcul care ajunge pe un nivel cât mai mic; dacă nu este necesar, din nodul respectiv nu mai ducem nici un arc.

Complexitatea algoritmului este $O(m \cdot \log m)$; este posibilă și obținerea unei complexități $O(m)$ dacă se folosesc liste de muchii pentru fiecare nod. Structura aleasă aduce avantaje deoarece poate fi vizualizată ușor, permițând o depanare eficientă, iar limitele problemei permit rezolvarea în complexitatea respectivă.

Listing CR7P1.PAS

```
Program Sens_Unic;
var i,j,k,l,m,n:Longint;
    fi,fo:Text;
    mm:array[1..6001,1..2] of Integer;
    sel:array[1..6000] of Integer;
    st,h,luate,nivel,t,thigh,p:array[1..501]
                                of Integer;

procedure readdata;
begin
  Assign(fi,'STRAZI.IN');
  Assign(fo,'STRAZI.OUT');
  Reset(fi);
  Readln(fi,n,m);
  for i:=1 to m do
  begin
    Readln(fi,mm[i,1],mm[i,2]);
    mm[m+i,1]:=mm[i,2];
    mm[m+i,2]:=mm[i,1]
  end;
  Close(fi)
end;

procedure Sort(l,r:Integer);
var i,j,x,y:Integer;
begin
  i:=l;
  j:=r;
  x:=mm[(l+r) div 2,1];
  repeat
    while mm[i,1]<x do i:=i+1;
    while x<mm[j,1] do j:=j-1;
    if i<=j then
    begin
      y:=mm[i,1];
      mm[i,1]:=mm[j,1];
      mm[j,1]:=y;
      y:=mm[i,2];
      mm[i,2]:=mm[j,2];
      mm[j,2]:=y;
      i:=i+1;
      j:=j-1
    end
  until i>j;
  if l<j then Sort(l,j);
  if i<r then Sort(i,r)
end;

procedure rec(k,niv:Integer);
var i,x:Integer;
begin
  if luate[k]=1 then Exit;
  luate[k]:=1;
  nivel[k]:=niv;
  h[k]:=niv;
  { folosim mai intai muchiile catre noduri
          care nu apartin inca arborelui }
  for i:=st[k] to st[k+1]-1 do
  begin
    x:=mm[i,2];
    if luate[x]=0 then
    begin
      t[x]:=k;
      rec(x,niv+1);
      if h[x]<h[k] then
        h[k]:=h[x]
    end
  end;
  { Mai avem nevoie de o muchie 'in sus'? }
  if h[k]<nivel[k] then p[k]:=1
  else thigh[k]:=t[k];
  { Actualizam valoarea h[k] folosind si }
  {    muchiile 'in sus' din k, pentru a }
  {    calcula corect h pentru parintele }
  { lui k. Daca avem nevoie de o muchie }
  {    'in sus', o alegem pe cea dorita }
  for i:=st[k] to st[k+1]-1 do
  begin
    x:=mm[i,2];
    if nivel[x]<nivel[k] then
    begin
      if x<>t[k] then
        if h[k]>nivel[x] then
        begin
          h[k]:=nivel[x];
          if p[k]=0 then
            thigh[k]:=x
        end
    end
  end
end;
```

```
procedure solve;
begin
  m:=2*m;
  sort(1,m);
  for i:=1 to n do
  begin
    st[i]:=1;
    while mm[st[i],1]<>i do
      Inc(st[i])
  end;
  st[n+1]:=m+1;
  { am construit structura de date:
       - mm - lista de muchii
       - st - pozitia in lista de la care
              incep vecinii fiecarui nod  }
  h[1]:=1;
  nivel[1]:=1;
  rec(1,1);
                                  { afisare }
  Rewrite(fo);
  for i:=1 to n do
    if t[i]<>0 then
      Writeln(fo,t[i],' ',i);
  for i:=1 to n do
    if thigh[i]>0 then
      Writeln(fo,i,' ',thigh[i]);
  Close(fo)
end;

Begin
  readdata;
  solve
End.
```

CR7P2: Munte

Linia orizontului dintr-un peisaj se poate desena pe un caroiaj, cu ajutorul caracterelor '/' și '\', aceasta conturându-se într-un "zig-zag" obișnuit.

O linie a orizontului este corectă dacă este desenată cu același număr de caractere '/', respectiv '\' și fiecare coloană a caroiajului conține un singur caracter. De asemenea, această linie trebuie să pornească dintr-o celulă a caroiajului aflată la nivelul mării și se termină tot într-o celulă aflată la nivelul mării.

Pe parcursul trasării, linia nu coboară niciodată sub nivelul mării. Linia orizontului începe cu caracterul '/' și se termină cu caracterul '\'.

După caracterul '/' poate fi plasat pe coloana următoare caracterul '/' pe linia superioară sau caracterul '\' pe aceeași linie.

Analog, după caracterul '\' poate fi plasat pe coloana următoare caracterul '\' pe linia inferioară sau caracterul '/' pe aceeași linie.

Definim vârful muntelui ca fiind format din două caractere '/\' situate pe aceeași linie și pe coloane alăturate.

Cerință

Scrieți un program care calculează numărul de posibilități de a trasa linii ale orizontului cu un număr dat de perechi de caractere '/' și '\', astfel încât să existe exact k vârfuri.

Date de intrare

Datele de intrare se vor citi din fișierul **MUNTE.IN** care va conține numerele n (numărul de caractere '/') și k (numărul de vârfuri) separate printr-un singur spațiu.

Date de ieșire

Rezultatul se va scrie în fișierul **MUNTE.OUT** care va conține numărul de posibilități de a trasa linii ale orizontului (având k vârfuri) folosind n perechi de caractere '/' și '\'.

Restricții

- $2 \le n \le 20$
- $1 \le k \le n$

Exemple

MUNTE.IN	MUNTE.OUT
2 2	1

MUNTE.IN	MUNTE.OUT
3 2	3

MUNTE.IN	MUNTE.OUT
4 1	1

MUNTE.IN	MUNTE.OUT
4 3	6

Soluție

Problema se rezolvă prin metoda programării dinamice. Se construiește matricea X unde $X[i, j, h, l]$ reprezintă numărul de munți construibili cu i caractere, având j vârfuri, cu ultimul caracter situat la înălțimea h; l indică tipul ultimului caracter (1 pentru '/', 2 pentru '\').

Relațiile de recurență sunt:

$X[i, j, h, '/'] = X[i - 1, j, h - 1, '/'] + X[i - 1, j, h, '\']$

$X[i, j, h, '\'] = X[i - 1, j - 1, h, '/'] + X[i - 1, j, h + 1, '\']$

Implementare: se folosesc matricele A și B, care reprezintă, pe rând, $X[i]$ și $X[i + 1]$. Matricele au ca elemente numere mari; în practică se folosesc pointeri către astfel de numere. Maniera de lucru cu aceste numere face programul mai clar, mai ușor de depanat și elimină problemele legate de limitele memoriei (numerele sunt alocate în *heap*).

Listing CR7P2.PAS

```
Program Munte;
type nr=record
          nc:Integer;
          a:array[1..100] of Byte
        end;
     numar=^nr;
```

```
var i,j,k,l,m,n,h:Longint;
    fi,fo:Text;
    a,b:array[0..21,0..21,1..2] of numar;
    x:numar;

procedure readdata;
begin
  Assign(fi,'MUNTE.IN');
  Assign(fo,'MUNTE.OUT');
  Reset(fi);
  Rewrite(fo);
  Readln(fi,n,k);
  Close(fi)
end;

procedure init(var a:numar);
begin
  if a=nil then
    New(a);
  Fillchar(a^,Sizeof(a^),0);
  a^.nc:=1
end;

procedure copy(a,b:numar);
begin
  init(a);
  a^:=b^
end;

procedure add(a,b:numar);
var i:Integer;
begin
  if b^.nc>a^.nc then
    a^.nc:=b^.nc;
  for i:=1 to a^.nc do
      a^.a[i]:=a^.a[i]+b^.a[i];
  for i:=1 to a^.nc do
  begin
    a^.a[i+1]:=a^.a[i+1]+a^.a[i] div 10;
    a^.a[i]:=a^.a[i] mod 10
  end;
  if a^.a[a^.nc+1]<>0 then
    Inc(a^.nc)
end;

procedure afis;
begin
  init(x);
  copy(x,a[k,1,2]);
  Rewrite(fo);
  for i:=x^.nc downto 1 do
    Write(fo,x^.a[i]);
  Writeln(fo);
  Close(fo)
end;
```

```
procedure solve;
begin
  for i:=0 to n do
    for j:=0 to n+1 do
      for l:=1 to 2 do
      begin
        init(a[i,j,l]);
        init(b[i,j,l])
      end;
  a[0,1,1]^.nc:=1;
  a[0,1,1]^.a[1]:=1;
  for i:=2 to n*2 do
  begin
    for j:=0 to i div 2 do
      for h:=1 to n do
      begin
        init(b[j,h,1]);
        init(b[j,h,2]);
        copy(b[j,h,1],a[j,h,2]);
        if h>1 then
          add(b[j,h,1],a[j,h-1,1]);
        copy(b[j,h,2],a[j,h+1,2]);
        if j>0 then
          add(b[j,h,2],a[j-1,h,1])
      end;
    for j:=0 to k do
      for h:=1 to n+1 do
        for l:=1 to 2 do
          a[j,h,l]^:=b[j,h,l]^
  end;
  afis
end;

Begin
  readdata;
  solve
End.
```

CR7P3: Mașina

O mașină trebuie condusă printr-un labirint din poziția de start până la poziția finală, ambele cunoscute, știind că viteza ei poate fi cel mult *K* și că după fiecare pas, viteza ei poate să se modifice cu 1. Mașina, într-o unitate de timp poate să se deplaseze doar pe orizontală sau verticală, iar atunci când se deplasează cu viteza *V*, parcurge o distanță egală cu *V*. În poziția de start mașina are viteza 0, iar în poziția de sosire trebuie să aibe viteza 1. Mașina poate să treacă de mai multe ori prin același punct.

Cerință

Scrieți un program care determină timpul minim necesar deplasării mașinii din punctul de start în poziția de sosire.

Date de intrare

Pe prima linie a fișierului de intrare **AUTO.IN** sunt scrise dimensiunile labirintului *N* (număr de linii) și *M* (număr

de coloane), precum și viteza maximă a mașinii (K), separate prin câte un spațiu. Fiecare din următoarele N linii conține fiecare exact M caractere, astfel încât spațiul reprezintă coridor, iar caracterul '*' reprezintă zid. Poziția de start este marcată cu litera 'S', cea finală cu litera 'F'.

Date de ieșire

În fișierul de ieșire **AUTO.OUT** se va scrie un singur număr, reprezentând timpul minim necesar pentru deplasarea mașinii din punctul de start în poziția finală. În cazul în care mașina nu poate să ajungă la destinație, în fișier se va scrie -1.

Restricții

- $1 \le N, M \le 100$
- $1 \le K \le 10$

Exemplu

AUTO.IN

```
5 10 3
**********
*S       *
****** ***
F
***   ****
```

AUTO.OUT

```
7
```

Soluție

Problema se rezolvă printr-o parcurgere în lățime. O situație în care se poate afla mașina este dată de (i, j, v), care reprezintă coordonatele și viteza.

Se introduce situația inițială într-o coadă, apoi se introduc situațiile adiacente ei ș.a.m.d.

Două situații se consideră adiacente dacă mașina poate trece la un pas din una în cealaltă (viteza diferă cu maxim 1, se încadrează în limite, pozițiile sunt pe aceeași orizontală/verticală și nu sunt despărțite de ziduri).

O situație introdusă în coadă se marchează în X pentru a nu fi introdusă din nou. $X[v, i]$ este mulțimea valorilor pentru j care corespunde situațiilor (i, j, v) în care se poate afla mașina.

Implementarea nu folosește coada, ci vectorul situațiilor accesibile la momentul T, din care se generează cel corespunzător momentului T+1.

Soluția este afișată la prima trecere prin situația finală; dacă aceasta nu se realizează și nu mai apar situații noi, se va afișa -1.

Programul poate fi optimizat, în special la testarea adiacenței a două situații însă, în forma actuală, este foarte ușor de depanat.

Listing CR7P3.PAS

```
Program Masina;
type ar=array[1..10000,1..3] of Integer;
var time,nr1,nr2,i,j,k,l,m,n,x1,y1,x2,y2:
                                    Longint;
    fi,fo:Text;
    s:array[1..102] of string[102];
    a,b:^ar;
    x:array[0..10,1..102] of set of 1..102;

procedure readdata;
begin
  Assign(fi,'AUTO.IN');
  Assign(fo,'AUTO.OUT');
  Reset(fi);
  Readln(fi,m,n,k);
  for i:=1 to m do
    Readln(fi,s[i+1]);
  Close(fi);
  for i:=1 to n+2 do
    s[1]:=s[1]+'*';
  s[m+2]:=s[1];
  for i:=2 to m+1 do
  begin
    s[i]:='*'+s[i];
    while Length(s[i])<n+1 do
      s[i]:=s[i]+' ';
    s[i]:=s[i]+'*'
  end
end;

procedure findx1y1;
begin
  for x1:=1 to m do
    for y1:=1 to n do
      if s[x1,y1]='S' then
        Exit
end;

procedure findx2y2;
begin
  for x2:=1 to m do
    for y2:=1 to n do
      if s[x2,y2]='F' then
        Exit
end;

procedure expand;
var ii,jj:Integer;
begin
  if (a^[i,1]=x2) and (a^[i,2]=y2) and
     (a^[i,3]=1) then
  begin
    Rewrite(fo);
    Writeln(fo,time);
    Close(fo);
    Halt
  end;
  for j:=a^[i,3]-1 to a^[i,3]+1 do
    if (j>0) and (j<=k) then
```

```
begin
  if a^[i,1]+j<m then
    if not (a^[i,2] in
                  x[j,a^[i,1]+j]) then
    begin
      l:=0;
      for ii:=1 to j do
        if s[a^[i,1]+ii,a^[i,2]]='*'
          then l:=1;
      if l=0 then
      begin
        Inc(nr2);
        b^[nr2,1]:=a^[i,1]+j;
        b^[nr2,2]:=a^[i,2];
        b^[nr2,3]:=j;
        x[j,b^[nr2,1]]:=x[j,b^[nr2,1]]+
                              [b^[nr2,2]]
      end
    end;
  if a^[i,1]-j>0 then
    if not (a^[i,2] in
                  x[j,a^[i,1]-j]) then
    begin
      l:=0;
      for ii:=1 to j do
        if s[a^[i,1]-ii,a^[i,2]]='*'
          then l:=1;
      if l=0 then
      begin
        Inc(nr2);
        b^[nr2,1]:=a^[i,1]-j;
        b^[nr2,2]:=a^[i,2];
        b^[nr2,3]:=j;
        x[j,b^[nr2,1]]:=x[j,b^[nr2,1]]+
                              [b^[nr2,2]]
      end
    end;
  if a^[i,2]+j<n then
    if not (a^[i,2]+j in
                  x[j,a^[i,1]]) then
    begin
      l:=0;
      for ii:=1 to j do
        if s[a^[i,1],a^[i,2]+ii]='*'
          then l:=1;
      if l=0 then
      begin
        Inc(nr2);
        b^[nr2,1]:=a^[i,1];
        b^[nr2,2]:=a^[i,2]+j;
        b^[nr2,3]:=j;
        x[j,b^[nr2,1]]:=x[j,b^[nr2,1]]+
                              [b^[nr2,2]]
      end
    end;
  if a^[i,2]-j>0 then
    if not (a^[i,2]-j in
                  x[j,a^[i,1]]) then
    begin
      l:=0;
      for ii:=1 to j do
        if s[a^[i,1],a^[i,2]-ii]='*'
          then l:=1;
      if l=0 then
      begin
        Inc(nr2);
        b^[nr2,1]:=a^[i,1];
        b^[nr2,2]:=a^[i,2]-j;
        b^[nr2,3]:=j;
        x[j,b^[nr2,1]]:=x[j,b^[nr2,1]]+
                              [b^[nr2,2]]
      end
    end
end;

procedure solve;
begin
  Inc(m,2); Inc(n,2);
  New(a);
  New(b);
  Fillchar(a^,Sizeof(a^),0);
  Fillchar(b^,Sizeof(b^),0);
  findx1y1;
  findx2y2;
  a^[1,1]:=x1;
  a^[1,2]:=y1;
  a^[1,3]:=0;
  nr1:=1;
  nr2:=0;
  x[0,x1]:=x[0,x1]+[y1];
  time:=0;
  while nr1>0 do
  begin
    for i:=1 to nr1 do expand;
    Inc(time);
    nr1:=nr2;
    nr2:=0;
    a^:=b^;
    Fillchar(b^,Sizeof(b^),0)
  end;
  Rewrite(fo);
  Writeln(fo,-1);
  Close(fo)
end;

Begin
  readdata;
  solve
End.
```

CONCURS de programare

Ediția a doua a concursului de programare organizat de Gazeta de Informatică începe în această lună. Pentru detalii referitoare la acest concurs citiți paginile următoare sau vizitați site-ul nostru: www.ginfo.ro.

Informații generale

Concursul de Programare Agora (*CPA*) este o competiție internațională destinată elevilor și studenților. Acest concurs este organizat de redacția ***GInfo*** și editura ***Computer Press Agora.***

Vor fi șase runde, la fiecare dintre ele participanții trebuie să rezolve trei probleme și să trimită soluțiile prin e-mail; adresa e-mail a acestui concurs este concurs@ginfo.ro. După cele șase etape care se vor desfășura prin e-mail va fi o etapă finală care se va desfășura la *Liceul de Informatică* din *Cluj-Napoca*; doar primii zece clasați în urma primelor șase runde vor participa la faza finală. Problemele care trebuie rezolvate sunt propuse de redacția *GInfo* în colaborare cu studenți și profesori care se ocupă pregătirea elevilor români care participă la diferite concursuri naționale și internaționale.

Înscrierea

Pentru a participa la *Concursul de Programare Agora* va trebui mai întâi să vă înscrieți. Pentru a vă înscrie va trebui să trimiteți un e-mail având următoarea structură:

```
From: numele dumneavoastră sau adresa e-mail
To: contest@ginfo.ro
Subject: înscriere APC
```

și în corpul mesajului:

```
Nume: numele dumneavoastră
Prenume: prenumele dumneavoastră
E-mail: adresa e-mail
Oraș: orașul în care locuiți
```

Vă vom trimite un e-mail care va conține codul dumneavoastră de identificare (ID).

Dacă adresa *CPA* nu va funcționa din diferite motive, puteți încerca și una dintre următoarele adrese:

- office@ginfo.ro
- clara@cs.ubbcluj.ro
- skortzy@email.ro
- smihai@asterix.obs.utcluj.ro

Regulament

Limbajele de programare acceptate la *CPA* sunt *Pascal* și *C/C++*. Fișierele sursă vor fi compilate folosind *Borland Pascal 7.0* sau *Borland C++ 3.1*.

Problemele vor apărea în paginile revistei *GInfo*, iar rezolvările vor trebui trimise înaintea termenului limită specificat în fiecare număr al revistei.

O singură soluție va fi acceptată pentru fiecare problemă. Dacă un concurent va trimite mai multe soluții pentru o anumită problemă numai prima soluție va fi luată în considerare, restul fiind ignorate.

Redacția *GInfo* își rezervă dreptul de a descalifica concurenții care trimit soluții identice (sau aproape identice).

Este interzisă folosirea de secvențe scrise în limbaj de asamblare; dacă un fișier sursă conține astfel de secvențe concurentul va primi 0 puncte pentru problema respectivă.

Este interzis ca programele generate folosind sursele concurenților să modifice timer-ul. Concurenții care vor încerca să modifice ceasul sistem vor fi descalificați.

Dacă aveți neclarități referitoare la enunțurile problemelor, puteți pune întrebări folosind adresa e-mail a concursului.

Nu puteți pune decât întrebări ale căror răspunsuri posibile sunt `DA` sau `NU`. Dacă răspunsul la întrebare pote fi găsit în enunțul problemei, atunci răspunsul la întrebare va fi `FĂRĂ COMENTARII`.

Dacă redacția *GInfo* consideră că răspunsul la o anumită întrebare trebuie primit de toți participanții atunci cel care a adresat întrebarea va primi răspunsul `ANUNȚ GENERAL`. Acest răspuns va fi urmat de un e-mail pe care îl vor primi toți participanții și care va clarifica nelămurirea.

Trimiterea soluțiilor

Pentru a trimite soluțiile de la o anumită rundă va trebui să trimiteți un e-mail având ca subiect codul dumneavoastră de identificare urmat de textul `Runda X`, unde `X` reprezintă numărul rundei (1-6). Acest e-mail va avea atașat o arhivă `.zip` sau `.rar` având numele `yyyyRx.ext` unde `yyyy` este codul dumnevoastră de identificare, `x` este numărul rundei, iar `ext` este extensia (`RAR` sau `ZIP`).

Arhiva va conține un singur director numit yyyy (codul dumneavoastră de înregistrare). Acest director trebuie să conțină patru fișiere: trei fișiere sursă corespunzătoare celor trei probleme ale rundei și un fișier .BAT care va conține liniile de comandă pe care doriți să le folosim pentru a vă compila sursele. Denumirile fișierelor sursă sunt date în descrierea problemelor, iar numele celui de-al patrulea fișier va fi ROUNDX.BAT. Din motive de securitate arhivele autoexpandabile nu sunt acceptate.

Arhiva nu poate conține nici un alt fișier sau director, deci nu puteți trimite fișiere auxiliare împreună cu fișierele sursă. Dacă programul dumneavoastră trebuie să creeze fișiere auxiliare atunci acestea vor avea extensia .TMP și trebuie șterse înaintea terminării execuției programului. Dacă această regulă nu este respectată concurentul va primi 0 puncte pentru problema corespunzătoare chiar dacă soluția este corectă. Dacă redacția *GInfo* consideră că un concurent pune în pericol (prin suprascrierea unor fișiere, crearea de fișiere foarte mari etc.) buna desfășurare a competiției atunci acel concurent poate fi descalificat.

Toate fișierele trimise devin proprietatea editurii *Computer Press Agora* și nu pot fi folosite în scopuri comerciale fără acordul scris al editurii.

Evaluarea

Pentru fiecare problemă puteți primi maximum 100 de puncte; numărul maxim de puncte pentru o anumită rundă este 300, iar pentru întreg concursul 1800. Pentru corectarea fiecărei probleme vom folosi între 5 și 20 seturi de date; dacă programul va genera un fișier de ieșire corect pentru un anumit set de date atunci concurentul va primi un număr de puncte care se va adăuga la numărul de puncte corespunzător problemei respective.

Calculatorul folosit pentru evaluare are următoarea configurație: procesor *Intel Pentium™ III 500 MHz*, RAM: 128 MB, HDD: 13 GB. Este garantat faptul că în timpul evaluării vor fi liberi cel puțin 300 KB de memorie convențională și cel puțin 100 MB de spațiu pe disc. Sistemul de operare folosit în timpul evaluării va fi *Microsoft Windows Me (Millennium Edition)*. Dacă un program va efectua o operație ilegală atunci concurentul va primi 0 puncte pentru setul de date corespunzător.

Sfaturi

Programatorii în *Pascal* nu ar trebui să folosească niciodată unitul CRT deoarece programele lor se vor opri returnând eroarea de execuție Error 200: Division by zero. Acest fapt este cauzat de un binecunoscut bug al mediului de programare *Borland Pascal 7.0*; nici o problemă propusă spre rezolvare în cadrul *Concursului de Programare Agora* nu necesită folosirea acestui unit.

Atunci când sunt folosite anumite tipuri de date (cum ar fi Comp) modul 8087 trebuie să fie activ. Opțiunile de mediu permit setarea acestui mod ca mod implicit, dar în mod normal acest mod nu este activ. Programatorii în *Pascal* sunt rugați să includă în codul sursă directiva de compilare {$N+} dacă doresc să folosească tipuri de date care necesită activarea acestui mod. În caz contrar, în mometul în care vom încerca să compilăm fișierul sursă, va fi generată eroarea de compilare Error 116: Must be in 8087 mode to compile this. Pentru a vă asigura că programul se va compila exact ca pe calculatorul folosit pentru a-l scrie, apăsați CTRL+O,O. Astfel, în sursa dumneavoastră, vor fi incluse toate directivele de compilare cu opțiunile care sunt setate în mediu. Puteți modifica aceste opțiuni astfel încât să fie cele dorite de dumneavoastră. Nu vă îngrijorați dacă nu cunoașteți semnificație unor anumite directive de compilare; acestea nu vor avea efecte negative asupra execuției programului - dacă programul dumneavoastră va rula corect pe calculatorul pe care l-ați folosit pentru a-l scrie, atunci va rula corect și pe calculatorul folosit pentru evaluare.

Toate programele trebuie să citească datele de intrare din fișierul de intrare specificat. Nu ar trebui să fie așteptate date de la intrarea standard (tastatură). Dacă programul dumneavoastră va aștepta o reacție de la tastatură atunci execuția sa va fi oprită după expirarea timpului și nu veți obține nici un punct pentru acea soluție chiar dacă ea este corectă. Deci, vă rugăm să nu includeți în codul sursă secvențe ca Readln, getch(), getche() etc.

Toate programele trebuie să scrie datele de ieșire în fișierul de ieșire specificat. Puteți scrie date și la ieșirea standard, dar aceasta nu va duce decât la încetinirea programului, deci nu este recomandabil să scrieți date pe ecran. Toate datele scrise la ieșirea standard sunt ignorate în timpul evaluării. Programul de evaluare va verifica numai soluția scrisă în fișierul de ieșire specificat.

Fișierul de intrare se află întotdeauna în directorul curent; nu încercați să citiți date din alte locații. Fișierul de ieșire va trebui generat tot în directorul curent; nu scrieți date de ieșire în alte locații. Trebuie să folosiți întotdeauna denumirile specificate în enunț pentru fișierul sursă, fișierul de intrare și fișierul de ieșire. Folosirea altor denumiri va duce la funcționarea incorectă a programului.

Nu trimiteți soluții după termenul limită (indicat în fiecare număr al revistei).

Exemplu

La fiecare rundă vor fi trei probleme care trebuie rezolvate. Să presupunem că una dintre ele are următorul enunț:

Nume fișier sursă: **SUMA.PAS, SUMA.C** sau **SUMA.CPP**
Nume fișier de intrare: **SUMA.IN**
Nume fișier de ieșire: **SUMA.OUT**
Descriere: Dându-se două numere, aflați suma lor.
Date de intrare: Fișierul de intrare conține, pe o singură linie, cele două numere separate printr-un singur spațiu.
Date de intrare: Fișierul de ieșire ar trebui să conțină suma celor două numere.
Exemplu:

SUMA.IN	SUMA.OUT
2 4	6

Timp maxim de execuție: 1 secundă/test

După citirea enunțului veți începe să rezolvați problema. Veți observa că nu este clar dacă numerele sunt întregi sau reale. Pentru a clarifica acest lucru va trebui să adresați o întrebare. Aceasta nu poate fi: *Numerele sunt întregi sau reale?* deoarece nu se poate răspunde la ea prin DA sau NU. O soluție ar fi să întrebați: *Numerele sunt întregi?* Veți primi răspunsul NU și veți ști că numerele sunt reale. Acum apare o întrebare mult mai importantă: *Ce precizie trebuie folosită pentru afișarea rezultatului?* Se pare că redacția *GInfo* a uitat să precizeze acest lucru în enunț. Bineînțeles, va trebui să adresați o întrebare la care să se poată răspunde prin DA sau NU. Ați putea încerca: *Este suficientă o precizie de trei zecimale?* Răspunsul la întrebare va fi ANUNȚ GENERAL și va fi urmat de un alt e-mail care va conține un mesaj de genul:

Rezultatul va trebui scris cu o precizie de două zecimale. Ne cerem scuze pentru descrierea incompletă a problemei.

Sperăm să nu fim niciodată nevoiți să trimitem astfel de mesaje, dar...

După primirea anunțului general puteți începe scrierea codului sursă. O soluție corectă ar putea fi:

```
 1: Program SUMA;
 2: var a,b:Real;
 3:     f:Text;
 4: Begin
 5:   Assign(f,'SUMA.IN');
 6:   Reset(f);
 7:   Readln(f,a,b);
 8:   Close(f);
 9:   Assign(f,'SUMA.OUT');
10:   Rewrite(f);
11:   Writeln(f,a+b:0:2);
12:   Close(f)
13: End.
```

Numerele de la începutul fiecărei linii nu fac parte din program; ele sunt folosite pentru a ne putea referi mai ușor la o anumită linie.

Dacă veți adăuga o linie `1': uses Crt;` și o linie `4': Clrscr;` nu veți reuși să obțineți decât eroarea `Error 200: Division by zero` așa cum am explicat în secțiunea *Sfaturi*.

Dacă adăugați o linie `12': Readln;` rezultatul evaluării va fi `Time out!` deoarece execuția programului nu se va termina, ci se va aștepta apăsarea tastei **<ENTER>**. Execuția va fi întreruptă de către programul de evaluare după o secundă.

Dacă folosiți linia `5: Assign(f,'C:\SUMA.IN');` atunci programul va funcționa corect numai dacă directorul curent este rădăcina drive-ului `C:` (ceea ce este foarte improbabil). Nu veți obține decât eroarea `File not found`.

Dacă folosiți linia `5: Assign(f,'ALTFISIER');` atunci programul nu va citi datele de intrare din fișierul specificat și va rula incorect.

Dacă folosiți linia `9: Assign(f,'C:\SUMA.OUT');` atunci programul va părea că funcționează corect dar programul de evaluare nu va găsi fișierul dumneavostră de ieșire decât dacă directorul curent este rădăcina drive-ului `C:` (din nou, șansele sunt foarte mici). Programul de evaluare va returna `No ouput file!`.

Dacă folosiți linia `9: Assign(f,'ALTFISIER');` atunci programul nu va scrie datele în fișierul de ieșire specificat, iar programul de evaluare va returna `No output file!`.

Dacă folosiți linia `7: Readln(a,b);` atunci programul va aștepta datele de intrare de la intrarea standard și va fi întrerupt după o secundă. Puteți folosi această linie numai dacă aveți o secvență asemănătoare cu:

```
5: Assign(Input,'SUMA.IN');
6: Reset(Input);
7: Readln(a,b);
8: Close(Input);
```

Dacă folosiți linia `11: Writeln(a+b:0:2);` atunci datele de ieșire vor fi scrise pe ecran și nu în fișierul de ieșire specificat. Puteți folosi această linie numai dacă secvența este de tipul:

```
 9: Assign(Output,'SUMA.OUT');
10: Rewrite(Output);
11: Writeln(a+b:0:2);
12: Close(Output)
```

Acestea sunt cele mai des întâlnite erori care au apărut la prima ediție a concursului organizat de *GInfo*. Sperăm că nu veți face aceleași greșeli și la această ediție.

După scrierea codului sursă trebuie să trimiteți soluția împreună cu soluțiile celorlalte două probleme. Să presupunem că denumirile fișierelor sursă pentru aceste probleme sunt **DIF.PAS** și **PRODUS.PAS**. Trebuie să creați arhiva pe care o veți trimite. Această arhivă va conține un director numit `yyyy` unde `yyyy` este numărul dumneavostră de identificare (de exemplu, `2457`). Veți crea directorul **2457** și veți copia în interiorul său fișierele **SUMA.PAS**, **DIF.PAS** și **PRODUS.PAS**. Acum trebuie creat fișierul .bat pe care îl vom folosi pentru compilarea surseor. Aceasta se va numi **RUNDA1.BAT** și ar putea arăta astfel:

```
bpc suma.pas
bpc dif.pas
bpc produs.pas
```

Folosiți `bpc` pentru compilarea surselor *Pascal* și `bcc` pentru compilarea surselor *C/C++*.

După crearea acestui fișier trebuie construită arhiva pe care o veți trimite. Folosiți un arhivator care creează arhive `.rar` sau `.zip`; nu creați o arhivă autoexpandabilă. Numele arhivei ar trebui să fie **2457R1.ZIP** sai **2457R1.RAR**.

Acum puteți trimite mesajul e-mail care va conține soluțiile. Nu uitați să atașați arhiva și să scrieți în linia *Subject*: `2457 Runda 1`.

Atenție!!!

Nu folosiți numărul 2457 decât dacă este propriul dumneavoastră cod de identificare pentru *Concursul de Programare Agora*. Folosiți ID-ul dumneavoastră; am folosit 2457 doar ca un exemplu.

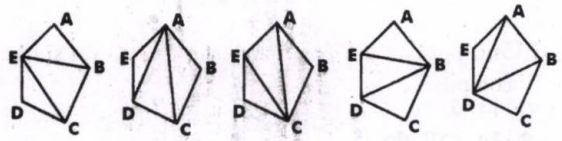
Runda 1

Vă prezentăm acum cele trei probleme de la prima etapă a Concursului de Programare Agora. Dacă doriți să trimiteți rezolvări pentru aceste probleme, nu uitați să vă înscrieți la concurs. Pentru ca rezolvările de la prima etapă să fie luate în considerare, va trebui să vă înscrieți până pe data de 5 Noiembrie 2000. Dacă vă veți înscrie după această dată, nu veți putea participa decât începând cu etapa a doua a concursului. Nu uitați să citiți regulamentul de participare și să țineți cont de sfaturile date de redacțiaGinfo. Mult succes tuturor participanților!

CPAR1P1: Triangularizare

Nume fișier sursă: `POLY.PAS`, `POLY.C` sau `POLY.CPP`
Nume fișier de intrare: `POLY.IN`
Nume fișier de ieșire: `POLY.OUT`
Descriere: Dându-se un poligon convex cu n laturi să se determine în câte moduri poate fi acesta triunghiularizat. Prin triangularizare se înțelege împărțirea poligonului în n - 2 triunghiuri sau trasarea a n - 3 diagonale care nu se intersectează decât în vârfurile poligonului.
Date de intrare: Fișierul de intrare conține un singur număr întreg n ($1 \le n \le 2457$) reprezentând numărul de laturi ale poligonului convex.
Date de ieșire: Fișierul de ieșire trebuie să conțină numărul triangularizărilor posibile.
Exemplu:

`POLY.IN`	`POLY.OUT`
5	5



Timp de execuție: 1 secundă/test

CPAR1P2: Divizori

Nume fișier sursă: `DIV.PAS`, `DIV.C` sau `DIV.CPP`
Nume fișier de intrare: `DIV.IN`
Nume fișier de ieșire: `DIV.OUT`
Descriere: Dându-se un număr natural nenul n să se determine care este cel mai mic număr natural care are exact n divizori.
Date de intrare: Fișierul de intrare conține un singur număr întreg n ($1 \le n \le 2457$) reprezentând numărul de divizori pe care trebuie să-i aibă numărul furnizat la ieșire.
Date de ieșire: Fișierul de ieșire trebuie să conțină un număr d care are exact n divizori.
Exemplu:

`POLY.IN`	`POLY.OUT`
6	12

Timp de execuție: 1 secundă/test

CPAR1P3: Premiere

Nume fișier sursă: `AWARD.PAS`, `AWARD.C` sau `AWARD.CPP`
Nume fișier de intrare: `AWARD.IN`
Nume fișier de ieșire: `AWARD.OUT`
Descriere: La un concurs trebuie premiați n participanți. Sponsorii au pus la dispoziția organizatorilor k categorii de premii, din fiecare categorie fiind disponibile oricâte premii. Determinați o modalitate de premiere a câștigătorilor astfel încât fiecare să primească un singur premiu și să fie oferit cel puțin un premiu din fiecare categorie.
Date de intrare: Fișierul de intrare conține două numere întregi n ($1 \le n \le 100$) și k ($1 \le k \le n$) reprezentând numărul de premianți, respectiv numărul de categorii de premii. Cele două numere sunt separate printr-un singur spațiu.
Date de ieșire: Fișierul de ieșire trebuie să conțină numărul posibilităților de premiere.
Exemplu:

`AWARD.IN`	`AWARD.OUT`
5 3	25

Timp de execuție: 1 secundă/test

Principiul INCLUDERII și EXCLUDERII

Doru Popescu Anastasiu

Principiul includerii și excluderii este unul dintre cele mai cunoscute rezultate din matematică. Acest articol vă propune să analizați modul în care acest principiu poate fi folosit pentru rezolvarea unor probleme de algoritmică.

Enunțul principiului

Fiind date $n \geq 2$ mulțimi finite $A_1, A_2, ..., A_n$ și notând cu $card(A)$ numărul de elemente al mulțimii A, avem:

$$card(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} card(A_i) - \sum_{i<j} card(A_i \cap A_j) + ...$$
$$+ (-1)^{n-1} card(\bigcap_{i=1}^{n} A_i)$$

Demonstrația acestui rezultat se face prin inducție matematică (vezi [2] sau [3]).

Aplicația 1

Se dau $p_1, p_2, ..., p_n$ ($n \leq 10$) numere prime distincte mai mici sau egale decât k ($k < 1.000.000.000$). Să se determine câte numere naturale mai mici sau egale cu k sunt divizibile cu unul din numerele $p_1, p_2, ..., p_n$.

Exemplu

Pentru $n=3$, $p_1=2$, $p_2=3$, $p_3=5$, $k=1000$ se va afișa: 734

Soluție

Considerăm mulțimile:

$$A_i = \{p_i q \mid q \in N, p_i q \leq k\}, i \in \{1, 2, ..., n\}.$$

Observăm că numărul căutat este

$$r = card(A_1 \cup A_2 \cup ... \cup A_n),$$

care este egal (conform principiului includerii și excluderii) cu:

$$\sum_{i=1}^{n} card(A_i) - \sum_{i<j} card(A_i \cap A_j) + ... + (-1)^{n-1} card(\bigcap_{i=1}^{n} A_i)$$

Folosind $card(A_i) = card(\{p_i q \mid q \in N, p_i q \leq k\}) = [k/p_i]$, $i \in \{1, 2, ..., n\}$, $card(A_i \cap A_j) = card(\{p_i p_j q \mid q \in N, p_i p_j q \leq k\}) = [k/(p_i p_j)]$, $i, j \in \{1, 2, ..., n\}$ etc. se obține:

$$r = \sum_{i=1}^{n} [k / p_i] - \sum_{i<j} [k/(p_i p_j)] + \sum_{i<j<k} [k/(p_i p_j p_k)] + ...$$
$$+ (-1)^{n-1} [k/(p_1 p_2 ... p_n)]$$

Un program care calculează acest număr este prezentat în continuare.

Listing APP1.PAS

```
Program Aplicatia_1;
var p:array[1..10] of Longint;
    n,i:Byte;
    k,r:Longint;

function suma(i:Byte):Longint;
var s:Longint;
    v:array[1..10] of Byte;
    h,q,j:Byte;
    pr:Real;
begin
  (* calculam cel de-al i-lea termen al
     sumei r, folosind metoda backtracking
     pentru a determina submultimile cu i
     elemente ale multimii {1,2,...,n}
  *)
  s:=0;
  h:=1;
  v[1]:=0;
  while h>0 do
  begin
    q:=0;
    while (q=0) and (v[h]<n) do
    begin
      Inc(v[h]);
      q:=1;
      if h>1 then
        if v[h]<=v[h-1] then
          q:=0
    end;
```

```
    if q=0 then Dec(h)
    else
      if h=i then
      begin
        pr:=k;
        for j:=1 to i do
          pr:=pr/p[v[j]];
        s:=s+Trunc(pr);
      end
      else
      begin
        Inc(h);
        v[h]:=0
      end
  end;
  suma:=s
end;                                    {suma}
Begin
  Write('n=');
  Readln(n);
  for i:=1 to n do
  begin
    Write('p[',i,']=');
    Readln(p[i])
  end;
  Write('k=');
  Readln(k);
  r:=0;
  for i:=1 to n do
    if i mod 2=1 then
      r:=r+suma(i)
    else r:=r-suma(i);
  Writeln(r)
End.
```

Aplicația 2

Se dă un număr natural nenul $n \le 14$. Să se determine numărul permutărilor mulțimii {1,2,...,n} *fără puncte fixe (adică permutări $p = (p_1, p_2, \ldots, p_n)$ cu proprietatea că $p_i \neq i$, pentru orice $i \in$* {1,2,...,n}*).*

Exemplu

Pentru $n = 3$ se va afișa: 2

Soluție

Notând cu A_i mulțimea celor $(n - 1)!$ permutări ale mulțimii {1,2,...,n} ce admit pe i ca punct fix, $i \in$ {1,2,...,n}, folosind principiul includerii și excluderii deducem:

$$card(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} card(A_i) - \sum_{i<j} card(A_i \cap A_j) + \ldots + (-1)^{n-1} card(\bigcap_{i=1}^{n} A_i)$$

Fie B o submulțime cu k elemente a mulțimii {1,2,...,n}. Atunci avem:

$$card(\bigcap_{i \in B} A_i) = (n-k)!$$

Fixând $k \le n$ există C_n^k astfel de submulțimi B. Rezultă:

$$card(\bigcup_{i=1}^{n} A_i) = C_n^1(n-1)! - C_n^2(n-2)! + \ldots + (-1)^{n-1} C_n^n.$$

Obținem astfel valoarea numărului căutat:

$$n! - C_n^1(n-1) + C_n^2(n-2)! - \ldots + (-1)^n C_n^n = \frac{n!}{2!} - \frac{n!}{3!} + \ldots + (-1)^n \frac{n!}{n!}.$$

Programul care determină această valoare este următorul:

Listing APP2.PAS

```
Program Aplicatia_2;
var nr,p:Longint;
    n,i,j:Byte;
Begin
  Write('n='); Readln(n);
  nr:=0;
  for i:=2 to n do
  begin {calculam raportul n!/i}
    p:=1;
    for j:=i+1 to n do p:=p*j;
      if i mod 2 =0 then nr:=nr+p
      else nr:=nr-p
  end;
  Writeln(nr)
End.
```

Aplicația 3

Se dau numerele naturale n și k cu $1 \le k \le 9$, $k \le n \le 11$. Să se determine câte numere compuse din n cifre există, care nu conțin decât cifrele 1, 2, ..., k, dar pe fiecare dintre acestea cel puțin o dată.

Exemplu

Pentru $n = 4$ și $k = 3$ se va afișa 36.

Soluție

Numărul total de numere de n cifre ce se pot forma cu cele k cifre 1, 2, ..., k este egal cu numărul de funcții de la o mulțime cu n elemente la alta cu k elemente și deci egal cu k^n. Fie acum A_i mulțimea numerelor de n cifre care nu-l conțin pe i, $i \in$ {1,2,...,k}. Rezultă:

$$card(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} card(A_i) - \sum_{i<j} card(A_i \cap A_j) + \ldots + (-1)^{n-1} card(\bigcap_{i=1}^{n} A_i)$$
$$= C_n^1(k-1)^n - C_n^2(k-2)^n - \ldots + (-1)^{k-1} C_k^{k-1}.$$

Deci numărul de numere compuse din n cifre, care nu conțin decât cifrele 1, 2, ..., k, dar pe fiecare dintre acestea cel puțin o dată este următorul:

$$k^n - C_k^1(k-1)^n + C_k^2(k-2)^n - \ldots + (-1)^k C_k^{k-1}.$$

Programul care calculează acest număr este următorul:

Listing APP3.PAS

```
Program Aplicatia_3;
var k,n,i:Byte;
    s:Longint;
```

```
function comb(p,q:Byte):Longint;
begin
  if (q=0) or (p=q) then comb:=1
  else comb:=comb(p-1,q)+comb(p-1,q-1);
end;

function prod(p,q:Byte):Longint;
var i:Byte;
    pp:Longint;
begin
  pp:=1;
  for i:=1 to q do pp:=pp*p;
  prod:=pp
end;

Begin
  Write('n='); Readln(n);
  Write('k='); Readln(k);
  s:=0;
  for i:=0 to k do
    if i mod 2=0 then
      s:=s+comb(k,i)*prod(k-i,n)
    else s:=s-comb(k,i)*prod(k-i,n);
  Writeln(s)
End.
```

Aplicația 4

Se dau n ($n \le 30000$) puncte în plan $P_1, P_2, \ldots, P_n$ și un număr natural k ($k \le 30000$). Să se verifice dacă se pot construi k segmente cu ambele capete în mulțimea formată din punctele $P_1, P_2, \ldots, P_n$, astfel încât să nu se formeze nici un triunghi cu vârfurile în aceste puncte. Dacă nu există soluție se va afișa mesajul `IMPOSIBIL`*. Dacă există soluție se va afișa o modalitate de construcție a segmentelor.*

Exemple

Pentru $n = 4$ și $k = 5$ se va afișa:

```
IMPOSIBIL
```

Pentru $n = 6$ și $k = 9$ o soluție este:

```
P1P3 P1P4 P1P5 P2P3 P2P4 P2P5 P6P3 P6P4 P6P5
```

Soluție

Pentru a rezolva această problemă avem nevoie de rezultatul următor:

Lemă

Fiind date n puncte în plan care se unesc între ele prin segmente, astfel încât să nu existe nici un triunghi cu vârfurile în cele n puncte, atunci există cel puțin un punct care este extremitatea a cel mult $[n/2]$ segmente (prin $[x]$ am notat partea întreagă a lui x).

Demonstrație

Pentru un punct x, fie A_x mulțimea punctelor legate cu x printr-un segment. Presupunem prin absurd că avem $card(A_x) \ge [n/2] + 1$, pentru orice $x \in X$, unde cu X am notat mulțimea celor n puncte din plan. Fie $x \in X$ și $y \in A_x$. Avem:

$$card(A_x \bigcup A_y) = card(A_x) + card(A_y) - card(A_x \bigcap A_y) \ge 2\left[\frac{n}{2}\right] + 2 - n.$$

deoarece $card(A_x \bigcap A_y) \le card(X) = n$.

Dar $2\left[\frac{n}{2}\right] + 2 - n = 2\left[\frac{n}{2}\right] + 1 - n + 1 \ge 1$, deci există

$$z \in A_x \bigcap A_y.$$

Am ajuns însă la o contradicție, deoarece x, y, z sunt vârfurile unui triunghi cu vârfurile în puncte din X. Deci există un punct $x \in X$ astfel încât $card(A_x) \le [n/2]$. Astfel demonstrația lemei este încheiată.

Revenind la problema noastră, vom demonstra că atunci când $k \le [n^2/4]$ există soluție, iar în cazul când $k \ge [n^2/4] + 1$, nu există soluție. Demonstrația o vom face prin inducție după n.

Pentru $n = 1$ sau $n = 2$ rezultatul este imediat, deoarece în primul caz numărul segmentelor este egal cu 0, iar în al doilea caz este egal cu 1.

Să presupunem afirmația adevărată pentru n puncte și să o demonstrăm pentru $n + 1$.

Din lema demonstrată anterior rezultă că există un punct $x \in X$ cu $card(X) = n + 1$, care este extremitatea a cel mult $[(n+1)/2]$ segmente. Dar mulțimea de puncte $X - \{x\}$ poate fi unită prin cel mult $[n^2/2]$ segmente astfel încât să nu se formeze nici un triunghi cu vârfurile în aceste puncte, deci numărul total de segmente care unesc punctele din mulțimea X (cu $n+1$ puncte) este majorat de $[(n+1)/2] + [n^2/4]$.

Vom demonstra în continuare că acest număr este egal cu $[(n+1)^2/4]$.

Avem două situații: n par și respectiv n impar.

- $n = 2h$, $h \in N$. Atunci:

$$[(n+1)/2] + [n^2/4] = h + h^2 = [(2h+1)^2/4] = [(n+1)^2/4],$$

deoarece este evident că

$$[(2h+1)^2/4] = [(4h^2 + 4h + 1)/4] = [h(h+1) + 1/4] = h(h+1).$$

- $n = 2h + 1$, $h \in N$. Atunci:

$$[(n+1)/2] + [n^2/4] = h + 1 + [(4h^2 + 4h + 1)/4] = h + 1 + h(h+1) = (h+1)^2 = [(n+1)^2/4].$$

Deci am arătat că:

$$[(n+1)/2] + [n^2/4] = [(n+1)^2/4].$$

Acum, în cazul în care $k \le [n^2/4]$, pebtru a obține exact k segmente de dreaptă între cele n puncte astfel încât să nu se formeze nici un triunghi cu vârfurile în aceste puncte, se poate proceda astfel: se grupează cele n puncte în două clase care conțin respectiv $[n/2]$ și $n - [n/2]$ puncte și se unesc numai puncte din prima clasă cu puncte din a doua clasă până se obțin k segmente.

Programul care rezolvă problema, este următorul:

Listing APP4.PAS

```
Program Aplicatia_4;
var n,i,k,h,j:Longint;
Begin
  Write('n='); Readln(n);
  Write('k='); Readln(k);
  if k>Trunc(n*n/4) then Write('IMPOSIBIL')
  else
    begin
      h:=0;
      for i:=1 to Trunc(n/2) do
        for j:=Trunc(n/2)+1 to n do
        begin
          Inc(h);
          if h<=k then Write('P',i,'P',j,' ')
        end
    end
End.
```

Bibliografie

[1] Brânzei D., Brânzei R., Anița S., Anița A., *Șiruri recurente în liceu*, Ed. GIL, Zalău, 1996

[2] Bușneag D., Maftei I., *Teme pentru cercurile și concursurile de matematică ale elevilor*, Ed. Scrisul Românesc, Craiova, 1983

[3] Ganga M., *Teme și probleme de matematică*, Ed. Tehnică, București, 1991;

[4] Sierpinski W., *Elementary Theory of Numbers*, Warszawa, 1987

Dl. prof. Doru Popescu Anastasiu este cadrul didactic la Colegiul Național "Radu Greceanu" din Slatina. Poate fi contactat prin e-mail la adresa dopopescu@pcnet.pcnet.ro.

GINFO Special Edition - CEOI 2000

Cu ocazia *Olimpiadei de Informatică a Europei Centrale* a apărut o ediție specială a revistei noastre, ediție dedicată acestui eveniment.

Datorită caracterului special al acestui număr care a fost destinat, în principal, participanților la CEOI 2000, această ediție a apărut în limba engleză.

GInfo Special Edition - CEOI 2000 a fost apreciată de toți participanții, mulți manifestându-și dorința de a avea în țara lor o revistă asemănătoare. Momentan, *Gazeta de Informatică* este singura revistă din lume care se adresează viitorilor specialiști în tehnologia informației, adică elevilor și studenților care sunt interesați de informatică.

Acest număr nu ar fi putut să apară fără ajutorul sponsorilor cărora le mulțumim și pe această cale. Cea de-a șaptea ediție a *Olimpiadei de Informatică a Europei Centrale* a fost sponsorizată de: *A&C International S.A.*, *Computer Press Agora*, *Computer Libris Agora*, *Societatea de Asigurare-Reasigurare ARDAF S.A.*, *Autodesk*, *Avitech*, *Net Brinel Computers*, *Cisco Systems*, *Compaq România*, *Comrace Computers*, *Connex - Xnet*, *Ericsson*, *Genesys*, *Forte Computers*, *Gazeta de Informatică*, *Hewlett Packard*, *IBM România*, *Lockheed Martin*, *Maguay*, *MB Distribution*, *Microsoft România*, *Minolta*, *Nextra*, *Pânda*, *RomTeleCom*, *Roxer*, *SCO*, *Softwin*, *Sysco*, *Tornado Systems*, *Turism Transilvania* și *Xerox*.

GInfo Special Edition - CEOI 2000 conține, printre altele, informații referitoare la cele 11 țări participante, enunțurile și soluțiile problemelor de la edițiile CEOI din 1998 (*Zadar*, *Croația*) și 1999 (*Brno*, *Republica Cehă*), enunțurile problemelor de la *Olimpiada Balcanică de Informatică* care s-a desfășurat anul acesta la *Ohrid* în *Macedonia*, interviuri cu domnul **Horia Georgescu**, președinte de onoare la CEOI 2000 și domnișoara **Anca Voicu**, director general la CEOI 2000 etc.

Din acest număr, tinerii informaticieni din *Europa* și nu numai, au aflat și de *Concursul de Programare Agora* care a devenit astfel primul concurs internațional de programare organizat de o revistă românească.

Mulțumită sponsorilor, *GInfo Special Edition - CEOI 2000* este distribuită gratuit, dar tirajul este limitat. Cititorii fideli care doresc să primească acest număr special pot contacta redacția *GInfo*, dar solicitările vor fi onorate în limita exemplarelor disponibile.

Redacția *GInfo* mulțumește tuturor colaboratorilor care au contribuit la realizarea acestui număr, în special celor care s-au ocupat de traducerile în limba engleză. Aceștia sunt *d-na prof. Georgeta Moldovan*, *d-na prof. Simona Motogna*, *Ada Ionescu*, *Răzvan Ristea* și *Adrian Monea*.

Nevoia de VITEZĂ

Diodor Bițan

Chiar dacă puterea de calcul a crescut foarte mult în ultima perioadă, nu toți sunt mulțumiți de performanțele calculatoarelor de astăzi. Este normal să dorim ca programele pe care le scriem să fie cât mai rapide. Acest articol prezintă câteva metode care pot fi folosite pentru accelerarea aplicațiilor pe care le proiectați.

Există tendința de a crede că datorită unor calculatoare mai performante, unor compilatoare bine realizate etc. programarea se va reduce în viitor la programare vizuală, folosind în principal mouse-ul. Este adevărat că tendințe reale în acest sens există, iar zilele bune ale programelor mici, optimizate până la cel mai mic detaliu, scrise 100% în assembler, fără nici un compromis privind viteza sau spațiul, se pare că au trecut. Dacă acum cinci ani orice program trebuia să încapă pe o dischetă, acum sunt destul de comune programele care au nevoie de mai multe CD-uri, gen *Microsoft Visual Studio* sau *Baldur's Gate*. În plus, conceptele ca portabilitatea, fiabilitatea, securitatea, lizibilitatea contează din ce în ce mai mult și împing vechile probleme ale vitezei și spațiului undeva în spatele scenei. Să ne resemnăm muncind cu ferestre în *Visual Basic*, cu baze de date unde toți algoritmii folosiți au fost deja implementați mult mai bine decât am putea noi să o facem vreodată? Să nu mai fie nevoie deloc de performanță? Să fie acesta sfârșitul?

Totuși lucrurile nu stau chiar așa; orice program care realizează ceva are nevoie de eficacitate. Nu fac excepție jocurile, compilatoarele, aplicațiile de grafică și sunet, arhivatoarele, sistemele de operare etc. Se spune că nu mai este nevoie de optimizări datorită implementărilor hardware (vezi acceleratoare grafice 3D). Dar implementarea hardware (care tot un fel de programare este) nu trebuie să o facă cineva? Fără să mai vorbim de driverele la care se lucrează continuu pentru câteva procente de viteză în plus. Poate că raportată la explozia calculatoarelor, a programelor prietenoase, a bazelor de date, a paginilor Web, pare în scădere, însă de fapt ***nevoia de viteză*** este din ce în ce mai mare și mult timp această nevoie nu va dispărea.

Ce merită optimizat?

Regula 10-90 (10% din program consumă 90% din timp), ne arată că este suficient să se mărească eficiența unor secțiuni critice ale codului sursă. Aceasta nu înseamnă că în restul programului se poate neglija total performanța. De exemplu, se poate ca o funcție să consume atât de mult timp încât celelalte pierderi de viteză să pară neglijabile. Dacă viteza de execuție a acestei funcții crește de zece ori la un moment dat, toate celelalte pierderi urcă de la unul, două procente la zeci de procente și trebuie și ele luate în considerare.

Pentru funcții de complexitate redusă, cu număr mic de variabile, este puțin probabil ca orice optimizare să crească viteza în mod esențial (*Microsoft Visual C++* memorează toate variabilele în regiștri și orice tentativă de îmbunătățire nu duce la rezultate spectaculoase). Pentru funcții complexe, cu multe variabile, cu multe condiții, orice optimizare care eliberează un registru se face simțită.

Pentru a obține performanță este foarte important să știm ce funcție consumă cel mai mult timp, să observăm efectele diferitelor schimbări, să putem verifica toate ideile pe care le găsim pe *Internet* sau în literatura de specialitate. Trecând peste metoda număratului (1000, 1001, ...) și a cronometrului, se poate folosi excelentul profiler al produsului *Microsoft Visual C++ 6.0*. Nu vom intra în detalii, dar rulând un program, se pot obține foarte ușor informații cum ar fi: numărul de apeluri pentru fiecare funcție, timpul consumat de execuția fiecărei funcții incluzând sau nu funcțiile apelate de aceasta, numărul de treceri prin fiecare linie de cod etc.

Folosirea funcțiilor `QueryPerformanceCounter()` și `QueryPerformanceFrequency()`, care returnează timpul cu o precizie mult mai bună decât funcțiile clasice de tipul `gettime()`, este o altă metodă care poate duce la creșterea performanțelor. Avantajul este că se pot face verificări în anumite puncte cheie din program, fără să scadă performanța.

Optimizarea funcțiilor

Parametrii nu trebuie să aibă dimensiuni mari; cu excepția câtorva cazuri, este mult mai bine ca parametrii să fie

transmiși prin referință și nu prin valoare. Această soluție este avantajoasă mai ales dacă se transmit clase ca parametri; în acest caz sunt apelați constructorii de copiere ai clasei respective (care pot să meargă foarte încet dacă, de exemplu, clasa respectivă alocă date dinamic în constructor).

Dacă suntem siguri că o funcție nu va modifica date la care nu ar trebui să aibă acces, se poate renunța la principiul încapsulării, protejării și accesului limitat la membrii claselor. Pentru funcții care se apelează des, sau cu parametri de dimensiuni mari, această optimizare este vitală.

```
typedef struct S{
  int a[50];
  char text[50];
};
class C{
public:
  int *data;
  int n;
  C(C& a){ // constructor de copiere
    data=new int[a.n];
    memcpy(data,a.data,n*sizeof(int));
  }
  ...
};
int Wrong(C a,C b,S s){...}
int Right(C& a,C* b,S* s){...}
```

Expandarea *inline* este o metodă de a anula costul apelurilor pentru funcții foarte des folosite în schimbul măririi dimensiunii fișierului executabil obținut în urma compilării și editării de legături. Oriunde apelăm o funcție *inline*, compilatorul înlocuiește apelul obișnuit al funcției (cu parametri copiați pe stivă și un "goto" spre funcție) prin intermediul codului funcției. Funcțiile se pot declara *inline* prin două metode:

```
inline int first(int a) {...} //prima medoda
class C {
  int second(int a) {...}
  // prin scrierea codului functiei chiar în
  // interiorul clasei, se subintelege
  // expandarea inline
};
```

Se pot seta opțiuni de compilare care să permită compilatorului să decidă ce funcții trebuie expandate *inline*. Simularea funcțiilor *inline* se poate realiza foarte bine și cu macrouri (care nici nu cer ca parametrii să fie de un tip anume):

```
#define MAX(a,b) (((a)>(b))?(a):(b))
```

Convențiile de apelare influențează ordinea și felul în care sunt transmiși parametrii. În cazul convenției *cdecl*, eliberarea stivei de parametri va fi realizată de către apelator ceea ce duce la creșterea dimensiunii fișierului executabil. Convențiile *stdcall* și *fastcall* plasează acest cod în interiorul funcției. În plus, *fastcall* transmite primii doi parametri în regiștri, mărind viteza. Funcțiile cu lungime variabilă de parametri vor fi apelate doar în convenție *cdecl*. (Doar apelantul cunoaște numărul parametrilor și poate "curăța" stiva corect. În momentul compilării nu se poate determina numărul parametrilor suplimentari care vor fi copiați pe stivă.)

```
__fastcall int printf(const char *format,
                      ...); // apel cdecl
__fastcall int scrie(char* txt);
                      // apel fastcall
```

În sfârșit, apelul funcțiilor virtuale consumă mai mult timp decât apelul celorlalte funcții membru ale unei clase, deoarece necesită o accesare suplimentară a *vtable*-ului clasei. Dacă nu știți cum se folosesc aceste funcții virtuale și nici nu vreți să învățați, acum aveți și un pretext pentru a vă ocupa de altceva.

Pointeri sau referințe

Cei care au încercat să scrie un joc sau un program mai mare, probabil au ajuns la un moment dat la surse care sunt pline de cod de genul:

```
void Xplode(int x,int y,int radius){
  for (int i=0;i<Game.nrMonsters;i++)
    if (abs(Game.monsters[i].x-x)+
          abs(Game.monsters[i].y-y)<radius)
      if (rand()%20>Game.monsterTypes[ \
          Game.monsters[i].type].defence[ \
        Game.monsters[i].experienceLevel]&&
                    !Game.monsterTypes[ \
              Game.monsters[i].type].XImmune)
        Game.monsters[i].DieByXplosion();
}
```

Pointerii sau referințele rezolvă problema indirectărilor excesive crescând atât lizibilitatea cât și viteza programului:

```
void Xplode(int x,int y,int radius){
  Monster *m;
  for (int i=0,m=Game.monsters[0];
                  i<Game.nrMonsters;i++,m++)
    if (abs(m->x-x)+abs(m->y-y)<radius){
      MonsterType &mt=Game.monsterTypes[ \
                                  m->type];
      if (rand()%20>mt.defence[ \
          m->experienceLevel] && !mt.XImmune)
        m->DieByXplosion();
    }
}
```

Pointerii pot fi folosiți și pentru accesarea mai eficientă a unei matrice:

```
int i,j,k;
int *pi;
for (j=0;j<100;j++)
  for (i=0;i<100;i++)
    m[j][i]++;
for (j=0;j<100;j++){
  pi=m[j];
  for (i=0;i<100;i++)
    (*pi++)++;
}
```

Compilatorul realizează aceste optimizări, în cazul ciclurilor necalculându-se adresa la fiecare accesare prin înmulțiri ci prin adunări repetate. Eficiența acestei optimizări relizate de compilator este mult mai mare în cazul prezentat anterior decât dacă scriem:

```
for (j=0;j<100;j++)
  for (i=0;i<100;i++)
    m[i][j]++; // schimbam m[j][i] cu m[i][j]
```

Se poate reduce numărul de regiștri folosiți ca să se acceseze variabile din memorie grupându-le în structuri:

```
void Wrong(int *a,int *b) {...}
  // se va folosi cate un registru pentru a
  // accesa a si b

typedef struct{
  int a[100];
  int b[100];
}comb;

void Right(comb *c) {...}
  // accesarea elementelor din a si b este
  // realizata folosind doar un registru
  // continand pointerul c
```

Același efect se obține în cazul claselor; prin intermediul pointerului **this** (care ocupă în general un registru al său) sunt accesate toate variabilele clasei.

O altă metodă de a mări viteza constă în folosirea variabilelor globale în punctele critice. Fiind alocate la compilare, aceste variabile nu sunt accesate prin parametri, ci direct, fără să se folosească regiștri suplimentari.

(Pre)calcule

Pe vremea când calculatoarele nu apăruseră și dinozaurii alergau liberi pe câmpii în deplină armonie cu universul, pentru a se realiza calculele complicate, se foloseau tăblițe de piatră pe care erau înscrise tabele cu valori ale diverselor funcții (sinus, logaritm etc.). Ideea în sine nu este rea și este uimitor de câte ori se poate folosi. Câștigul în viteză este foarte mare și prețul memoriei suplimentare necesare tabelelor și precizia ce scade destul de mult nu cântăresc prea mult în balanță. Probabil că ați admirat screen saverul cu lupa ce se plimbă pe ecran. Acesta făcea toate calculele geometrice la început (mesajul "*Rendering Lens*") și rezultatul era o matrice pătrată în care fiecare căsuță conținea coordonatele *x* și *y* ale pixelului ce trebuie luat de pe display și pus în punctul respectiv. Astfel se pot realiza deformări oricât de exotice și de costisitoare.

Problema cu tabelele de valori este că, având doar valori discrete, precizia scade nepermis de mult. Parte din precizie se poate recupera făcând o simplă interpolare liniară:

```
float f_buf[1000]; // valori ale unei functii
                   // dificil de calculat
...
float f(float x){ // presupun x intre 0,998
  int i=x; // partea intreaga a lui x
  float fr=x-i; // partea fractionara a lui x
  return f_buf[i]*(1-fr)+f_buf[i+1]*fr;
  // nu se foloseste nici o impartire pentru
  // ca intervalul dintre doua valori
  // consecutive ale functiei este 1
}
```

Precalculările se pot folosi și fără eroare de precizie și fără memorie suplimentară, uneori producând accelerări radicale, ca în următorul exemplu, o funcție care realizează rotirea unor vectori 2D:

```
void slowRotate(float alpha){
  for (int i=0;i<n;i++){
    vf[i].x=vi[i].x*cos(alpha)+
                          vi[i].y*sin(alpha);
    vf[i].y=-vi[i].x*sin(alpha)+
                          vi[i].y*cos(alpha);
  }
}

void fastRotate(float alpha){
  float s=sin(alpha),c=cos(alpha);
  for (int i=0;i<n;i++){
    vf[i].x=vi[i].x*c+vi[i].x*s;
    vf[i].y=-vi[i].x*s+vi[i].x*c;
  }
}
```

Creșterea mare de viteză apare datorită faptului că `sin()` și `cos()` sunt funcții.

Includem tot la precalculări și următoarea optimizare:

```
for (q=0;q<100;q++)
  v[q]=q*q; // 0 1 4 9 16 ...
for (q=0,w=0;q<100;++q,w+=q*2-1)
  v[q]=w; // 0 1 4 9 16 ...
```

Optimizări cu numere întregi

De ce este de 10 ori mai greu să calculăm 12934534 împărțit la 9 decât 12934534 împărțit la 10? Pentru că oamenii preistorici, având și ei 10 degete, ne-au transmis sistemul zecimal. (Mare ghinion: dacă ar fi avut 16 degete am fi colonizat planeta *Marte* acum 200 de ani; ce s-ar fi întâmplat dacă am fi avut 256 de degete?) Tot așa, calculatorului (care folosește sistemul binar) îi este foarte ușor să împartă și să înmulțească cu puteri ale lui 2, operații cunoscute sub numele de *shift*-ări ce constau în deplasări de biți la dreapta (înmulțiri cu 2, operator <<) sau la stânga (împărțiri la 2, operator >>):

```
i=5;    // 000101 5
i<<=2; // 010100 20    i=i*4;
i>>=3; // 000010 2     i=i/8;
```

Chiar dacă compilatoarele realizează aceste optimizări automat, cei care le cunosc vor căuta să dea compilatorului cât mai multe ocazii de a le pune în aplicare. De exemplu, renunțând la sistemul cu 360 grade și adoptând unul cu 256 grade. Sau având grijă ca matricele să aibă ca dimensiuni puteri ale lui doi:

```
int a[256][256];
...
for (int x=0;x<255;x++)
  for (y=0;y<255;y++)
    a[y][x]++;
    // viteza nu creste. Compilatorul ac-
    // tualizeaza pointerul a[y][x] printr-o
    // simpla adunare la fiecare trecere

int getxy(int x,int y){
  return a[y][x];
}
  // adresa de memorie se calculeaza foarte
  // usor, printr-o shift-are viteza este mai
  // mare decat in cazul cand dimensiunile
  // matricei nu erau puteri ale lui 2
```

Se pot realiza înmulțiri și cu numere diferite de puterile lui 2, mai rapide decât înmulțirile normale. Exemplul clasic este cel de `putpixel()` pentru modul 13h

```
char* video_memory=(char*)0xa0000;
void slowPutPixel(int x,int y,char col){
  video_memory[y*320+x]=col;
}
#define fasterPutPixel(x,y,col){
  video_memory[(y)<<6+(y)<<8+(x)]=col;
}
```

Este evident că `y<<6+y<<8=y*64+y*256=y*320`. Orice înmulțire poate fi calculată folosind un astfel de artificiu. De exemplu, o înmulțire cu 255 poate fi efectuată astfel:

```
#define times255(x) ((x)<<8-(x))
  // x<<8-x=256*x-x=255*x
```

Se poate de asemenea accelera operația de modul cu numere puteri ale lui doi. Această operație este echivalentă cu păstrarea ultimilor biți dintr-un număr, care se poate realiza cu un `SI` pe biți: `i%=16` este echivalentă cu `i&=(15)`. Avem $15=00001111_{(2)}$, deci operația este corectă.

Operații cu virgulă fixă

O altă opinie este aceea că viteza de efectuare a operațiilor în virgulă mobilă s-a îmbunătățit atât de mult încât folosirea ei nu duce la o încetinire importantă a execuției programului. Într-adevăr, viteza scade doar de câteva ori, dacă folosim **`float`** pentru a păstra numere la care interesează doar o zecimală exactă. Alternativa o reprezintă virgula fixă, care de multe ori este preferabilă. Vom da un exemplu de virgulă fixă pentru 16 biți din care 8 sunt folosiți pentru cifrele zecimale. Principalele operații pot fi implementate astfel:

```
typedef struct{
  unsigned char fract;
    // partea fractionara a numarului
  signed char deci;
    // partea zecimala a numarului
} smart;

typedef union{
  smart d;
  __int16 i;
} fixed_16;

#define ORI_F16(x,y) ((__int32)(x)*(y)>>8)
#define IMP_F16(x,y) (((__int32)(x)<<8)/(y))
#define TOFLOAT_F16(x) ((x)/256.0)

void test(){
  fixed_16 a={128,1},b={128,2},c={64,20},d;
    // a=1.5, b=2.5, c=20.25
  printf("(a+b)/(b*c) = %f",TOFLOAT_F16(
      IMP_F16((a.i+b.i),ORI_F16(b.i,c.i))));
  if (a.i>b.i)
    puts("a greater than b\n");
                              // comparatia.
}
```

Se observă că singura pierdere de viteză față de înmulțirile și împărțirile numerelor întregi este o *shift*-are. Se pot scrie în assembler operațiile, și pentru tipuri de date pe mai mulți biți și cu poziția virgulei oriunde, dar se găsesc pe *Internet* biblioteci care să facă toate acestea.

Loop-ul cel rău

Pentru a nu păți ca în povestea de tristă amintire, *loop*-ii noștri trebuie să se miște cât mai repede. Să considerăm

următorul ciclu folosit pentru a aduna în vectorul *c* conținutul vectorilor *a* și *b*.

```
int a[100],b[100],c[100];
int k=100;
for (int i=0;i<k;i++)
  c[i]=a[i]+b[i];
```

O construcție echivalentă este:

```
i=0;
goto loop_start;
loop_condition:
i++;
if (i>=k)
  goto loop_end;
    // la compilare devine jmp conditionat
loop_start:
c[i]=a[i]+b[i];
goto loop_condition; // jmp neconditionat
loop_end:
```

Se observă că în loc de un `jmp` condiționat și un `jmp` necondiționat, se poate face doar `jmp`-ul condiționat, înlocuind ciclul **for** cu unul **do while**:

```
i=0;
do{
  c[i]=a[i]+b[i];
  i++;
}while (i<k);
```

Am presupus că avem $k > 0$. Se poate scăpa de comparația din condiție și deci de accesarea de 100 de ori a variabilei *k* (sau ocuparea unui registru) inversând ordinea de parcurgere a ciclului:

```
i=k;
do{
  i--;
  c[i]=a[i]+b[i];
}while (i);
```

Dacă *k* este divizibil cu doi, se poate reduce la jumătate numărul de comparații și salturi desfășurând ciclul (*"loop unrolling"*):

```
i=k;
do{
  c[--i]=a[i]+b[i];
  c[--i]=a[i]+b[i];
}while (i);
```

S-ar fi putut repeta de *n* ori expresia din interiorul ciclului, dar *k* ar fi trebuit să fie divizibil cu *n*. În cazul în care *k* nu este neapărat divizibil, și considerăm $n=8$, se poate folosi un artificiu numit *"Duff's Device"* care este ceva între magie și artă. Vă lasăm plăcerea de a afla singuri cum și de ce acest algoritm funcționează:

```
switch ((i=k)&7){
  // (i=k)&7 calculeaza modulul impartirii lui
  // k la 8 si i=k
  case 0: do{ c[--i]=a[i]+b[i];
           case 7:c[--i]=a[i]+b[i];
           case 6:c[--i]=a[i]+b[i];
           case 5:c[--i]=a[i]+b[i];
           case 4:c[--i]=a[i]+b[i];
           case 3:c[--i]=a[i]+b[i];
           case 2:c[--i]=a[i]+b[i];
           case 1: c[--i]=a[i]+b[i];
         }while (i);
}
```

Desfășurarea ciclului se poate folosi cu și mai mult succes pentru a elimina total unele cicluri. De exemplu, funcția următoare stabilește numărul de vecini aflați pe verticală, orizontală sau diagonală care au valoarea 1 într-o matrice ale cărei elemente au valori 0 și 1.

```
int nVec(int x,int y){
  int n=0;
  for (int i=-1;i<=1;i++)
    for (int j=-1;j<=1;j++)
      if (i || j)
        n+=matrix[x+i][y+j];
  return n;
}
```

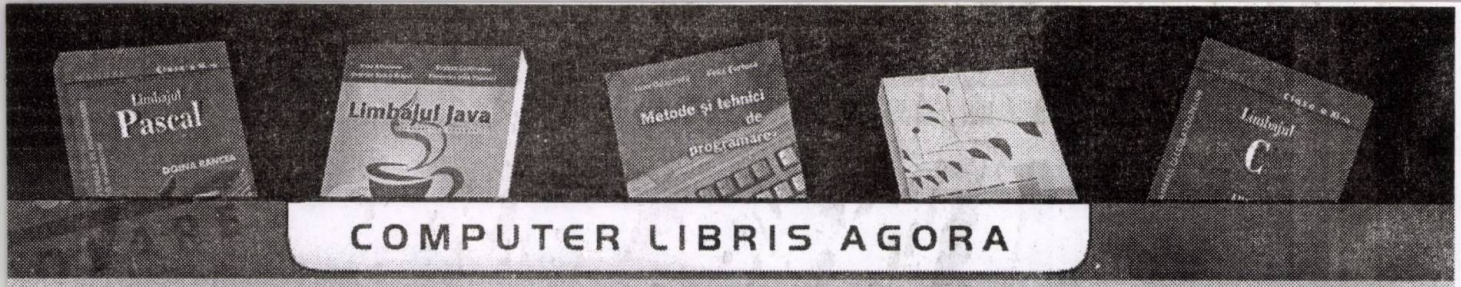
Funcția se poate scrie și astfel:

```
int nVec(int x,int y) {
return matrix[x+1][y+1]+
  matrix[x+1][y]+
  matrix[x+1][y-1]+
  matrix[x][y+1]+
  matrix[x][y-1]+
  matrix[x-1][y+1]+
  matrix[x-1][y]+
  matrix[x-1][y-1];
}
```

Și ultima accelerare sau ***accelerarea ultimă***, care funcționează pentru orice program sau componentă hardware, chiar și pentru *Windows '95*, este $g = 9.8\ m/s^2$.

Ne bucurăm că acest articol nu a suferit încă această ultimă optimizare (ați ajuns până la sfârșit), vă urăm un călduros *live long and prosper*.

Diodor Bițan este student în anul III la Universitatea Politehnică din București. Poate fi contactat prin e-mail la adresa b_diodor@hotmail.com.



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

SETI@home

Mihai Scorțaru

Mulți cred că există viață în spațiul extraterestru, dar nimeni nu a reușit niciodată să demonstreze că nu suntem singuri în Univers. Există câteva proiecte științifice care încearcă să determine dacă există inteligență extraterestră. Unul dintre aceste proiecte este SETI (Search for ExtraTrerrestrial Intelligence - Căutarea Inteligenței Extraterestre); acest articol prezintă modul în care puteți ajuta la definitivarea acestui proiect.

Există multe metode folosite pentru căutarea inteligenței extraterestre; multe dintre ele constau în cercetarea a miliarde de frecvențe radio care inundă Universul căutând semnale care ar putea fi transmise de ființe inteligente. O altă metodă este de a căuta semnale în pulsurile de lumină emanate de către stele.

Unul dintre proiectele *SETI* este *SETI@home*; acest proiect permite oricărei persoane care posedă un calculator și o legătură *Internet* să participe la căutare. Calculatorul este folosit numai atunci când utilizatorul nu este prezent, deci activitatea sa nu este afectată. Este instalat un screen-saver și, în timp ce acest screen-saver rulează, calculatorul caută inteligență extraterestră.

Despre SETI

Cu siguranță majoritatea cititorilor au vizionat filme ca *Star Trek*, *Războiul Stelelor*, *Alien*, *Ziua Independenței* etc. Toate aceste filme au ca principal subiect civilizațiile extraterestre și relațiile dintre acestea și societatea umană. Uneori extratereștrii sunt prietenoși, alteori sunt ostili, dar o caracteristică predominantă este aceea că sunt umanoizi.

Mulți oameni speră că, în viitorul apropiat, se va realiza un contact între umanitate și o civilizație extraterestră. În acel moment vom avea o mulțime de lucruri de aflat și învățat unii de la alții. *SETI* este unul dintre proiectele care încearcă să facă acest moment posibil și ajutorul vostru ar fi apreciat.

Dacă presupunem că extratereștrii încearcă să ne contacteze atunci ar trebui să căutăm smnalele lor. Există mai multe proiecte care încearcă să găsească dovada vieții în cosmos. Aceste proiecte poartă numele colectiv *SETI* (*Search for ExtraTerrestrial Intelligence - Căutarea Inteligenței Extraterestre*).

Galaxii îndepărtate

Civilizații extraterestre

Galaxia noastră are peste 400 de miliarde de stele, *Soarele* nostru fiind doar una dintre ele. Universul are miliarde de galaxii, iar *Calea Lacteae* este doar una dintre ele. S-ar părea că ar trebui să existe o mulțime de ființe inteligente în Univers. Vom încerca să facem o estimare a numărului de civilizații extraterestre din galaxia noastră. Primul care a încercat să obțină o astfel de aproximare a fost astronomul ***Frank Drake***. O simplă ecuație, cunoscută acum sub numele de ***ecuația Drake***, ne arată cam care ar fi posibilitățile. Ecuația este destul de ușor de înțeles, deci nu trebuie să fiți genii pentru a putea citi paragrafele următoare. Formula este:

$N = R \cdot f(p) \cdot n(e) \cdot f(l) \cdot f(i) \cdot f(c) \cdot L$

Vom explica acum semnificația variabilelor care apar în ecuație:

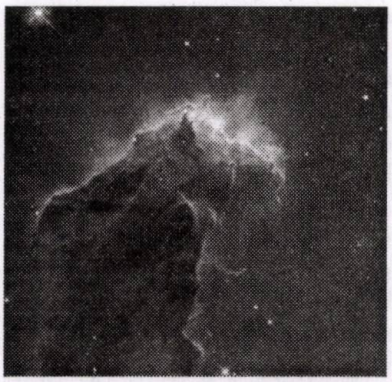
- N reprezintă numărul de civilizații din *Calea Lacteae* care au posibilitatea de a comunica
- R reprezintă rata de formare a stelelor din galaxia noastră
- $f(p)$ reprezintă fracțiunea stelelor în jurul cărora se învârt planete
- $n(e)$ reprezintă fracțiunea planetelor care se află în interiorul ecosferei stelei în jurul căreia se învârt; o ecosferă este zona din jurul unei stele care permite apariția condițiilor favorabile vieții
- $f(l)$ reprezintă fracțiunea planetelor din ecosferă pe care se dezvoltă viața
- $f(i)$ reprezintă fracțiunea planetelor pe care s-a dezvoltat viața și au apărut și ființe inteligente
- $f(c)$ reprezintă fracțiunea planetelor pe care ființele inteligente au dezvoltat o tehnologie și au încercat să comunice
- L reprezintă "durata de viață" a unei civilizații care a dezvoltat o tehnologie pe care o poate folosi pentru a comunica, adică perioada de timp care trece din momentul în ca-

re o civilizație este capabilă să comunice și momentul în care aceasta va dispărea.

În paragrafele următoare vom prezenta mai detaliat fiecare dintre factorii care apar în *ecuația Drake* și vom încerca să le atribuim valori rezonabile.

Formarea planetelor

Chiar dacă rata formării stelelor a fost mult mai mare atunci când s-a format galaxia noastră, este evident faptul că stele se nasc și astăzi. O aproximare acceptabilă pentru rata de formare a stelelor este de 20 stele/an, deci avem $R = 20$.



Nașterea unei stele în nebuloasa Vulturului

Stele cu planete

Stelele se formează din nori uriași de gaze care colapsează. Mulți astfel de nori posedă o mișcare de rotație; pe măsură ce colapsează ei se rotesc tot mai repede și se formează un disc de gaz. În centru va apărea steaua propriu-zisă și, spre exterior, pot exista vârtejuri mai mici de gaz care pot forma planete. Până recent nu a existat nici o dovadă a prezenței de sisteme planetare în jurul altor stele în afara *Soarelui*. În ultimii anii mai multe echipe de astronomi au anunțat descoperiri de planete în jurul stelelor apropiate. Aceste extraordinare descoperiri ne fac să credem că posibilitatea existenței de planete în jurul a multe stele este mult mai mare decât credeam acum câțiva ani. Putem estima că jumătate dintre stele formează sisteme planetare, iar cealaltă jumătate formează sisteme binare de stele. În concluzie, putem considera că $f(p)$ are valoarea 0,5.

Planete în ecosferă

Factorul $n(e)$ este puțin mai special și este destul de greu să găsim pentru el o valoare apropiată de cea reală.

Stelele mici sunt reci și roșii; planetele lor ar trebui să orbiteze foarte aproape pentru a fi în ecosferă. Ecosferele acestor stele sunt foarte subțiri, deci nu prea există loc de planete în interiorul lor. Mai mult, planetele care orbitează foarte aproape de stea au durata mișcării de rotație egală cu cea a mișcării de revoluție, deci o față este permanent luminată de stea, iar cealaltă față este permanent întunecată (situația este similară în cazul *Pământului* și a *Lunii*, doar o față a satelitului fiind vizibilă de pe planetă). Atmosfera acestei planete va îngheța pe partea întunecată; aceste condiții nu sunt favorabile vieții.

Pe de altă parte stelele mari și albastre au o temperatură foarte ridicată, deci și o ecosferă mai îndepărtată și mai groasă. Totuși, judecând din punctul de vedere al sistemului nostru solar, planetele sunt mai depărtate unele de altele pe masură ce sunt mai depărtate de stea, deci mărimea ecosferei este anulată de acest efect. De asemenea, stelele mari își consumă combustibilul foarte repede, deci durata lor de viață este foarte scurtă. De obicei aceste stele devin nove sau supernove înainte ca viața să aibă șansa de a apărea.

Soarele nostru este o stea de mărime medie și are două (*Pământ* și *Marte*) sau poate trei (+*Venus*) planete în interiorul ecosferei.

O aproximare acceptabilă pentru numărul de planete din interiorul "zonei de viață" este $n(e) = 1$.

Viața pe planete

Cea mai mare dificultate care apare la estimarea factorului $f(l)$ este aceea că avem doar câteva exemple de planete unde condițiile sunt favorabile apariției vieții. Așa cum am afirmat anterior, este posibil ca *Venus*, *Pământ* și *Marte* să fi avut condiții favorabile. Știm că viața a evoluat pe *Pământ* și există dovezi din ce în ce mai clare care ne fac să credem că și pe *Marte* au existat forme de viață primitive cu miliarde de ani în urmă. O aproximare acceptabilă pentru acest factor este 0,2; cu alte cuvinte, pe una din cinci planete care oferă condiții favorabile vieții, viața va apărea.

Viața inteligentă

Este foarte dificil să determinăm pe câte dintre planetele pe care a apărut viața, aceasta va evolua până la apariția inteligenței.

Totuși, dacă acceptăm dovezile oferite de teoria selecției naturale, a supraviețuirii celui mai bine adaptat, s-ar părea că există 100% șanse ca viața inteligentă să apară, deoarece inteligența este un "produs" natural al evoluției.

Singurul exemplu pe care îl avem la dispoziție este *Pământul*, deci nu putem decât să presupunem că $f(i) = 1$.

Societăți tehnologice

Trebuie să vedem acum câte dintre speciile inteligente vor dezvolta o tehnologie și o vor folosi pentru a comunica. Pe *Pământ* oamenii au realizat acest lucru, dar balenele și delfinii nu au dezvoltat niciodată o tehnologie chiar dacă au un anumit grad de inteligență. Ca o primă aproximare vom atribui factorului $f(c)$ valoarea 0,5.

Autodistrugerea

Factorul cel mai greu de determinat este L; el indică cât timp va supraviețui o civilizație care este adeptă a tehnologiei și o folosește pentru a comunica. Umanitatea se află în această fază a evoluției de aproximativ 50 de ani. Apare în mod firesc întrebarea dacă civilizațiile avansate se autodistrug după ce dezvoltă o tehnologie care să permită acest lucru sau își dau seama că problemele apărute între membrii societății trebuie rezolvate înainte ca disputele între diferite facțiuni să ducă la autodistrugere. Pentru moment nu vom atribui nici o valoare acestui termen.

Rezultatul final

Vom înlocui termenii din *ecuația Drake* cu valorile alese:

$N = R \cdot f(p) \cdot n(e) \cdot f(l) \cdot f(i) \cdot f(c) \cdot L$,
adică

$N = 20 \cdot 0{,}5 \cdot 1 \cdot 0{,}2 \cdot 1 \cdot 0{,}5 \cdot L$

Obținem $N = L$, deci numărul de civilizații capabile să comunice din galaxia noastră este egal cu numărul de ani care trec de la dezvoltarea tehnologiei până la distrugerea sau autodistrugerea civilizației.

Concluzii

Majoritatea oamenilor de știință speră că dacă o civilizație reușește să treacă peste impulsul inițial de a se autodistruge folosind propria tehnologie, atunci este foarte probabil ca acea civilizație să supraviețuiască o perioadă lungă de timp.

Totuși, ar trebui să existe cel puțin 50 de civilizații în *Calea Lacteae* (numărul de ani pe care omenirea i-a petrecut folosind tehnologia pentru a comunica), iar dacă o civilizație capabilă să comunice poate supraviețui milioane de ani, s-ar putea să existe chiar milioane de civilizații pe care le putem căuta.

Valorile pe care le-am indicat în paragrafele anterioare sunt cele propuse de către *SETI*. Totuși, în opinia noastră, unele valori au fost mărite artificial. De exemplu, este foarte probabil ca *SETI* să exagereze afirmând că $f(l) = 0{,}2$; probabil că valoarea reală este mult mai apropiată de 0.

Nimeni nu poate însă afirma că nu există nici o posibilitate ca civilizații extraterestre să încerce să ne contacteze. Sperăm că am reușit să vă convingem că programele *SETI* nu sunt doar o pierdere de timp.

Despre SETI@home

Datele furnizate de un telescop trebuie să fie procesate pentru a putea cerceta prezența unor dovezi care să indice existența inteligenței extraterestre. Au fost construite supercalculatoare destinate acestui proiect, dar datorită volumului uriaș de date care trebuie procesate, aceste supercalculatoare nu cercetează în amănunțime toate aceste date, ele neverificând prezența unor semnale slabe. De asemenea, anumite tipuri de semnale nu pot fi cercetate de aceste supercalculatoare. Principalul motiv care a dus la această situație este acela că puterea de calcul care este disponibilă pentru analiza datelor este limitată.

Echipa *SETI* a descoperit că există mii de calculatoare care ar putea fi folosite, dar nimeni nu și-a dat seama încă. Multe calculatoare afișează pe ecran screen saver-e nerealizând practic nimic și irosind energie electrică. Aici este locul în care *SETI@home* apare pe scenă. Proiectul *SETI@home* încearcă să vă convingă să permiteți echipei *SETI* să vă împrumute calculatorul atunci când nu îl folosiți. În acest fel veți putea ajuta la "*...căutarea de noi forme de viață și de noi civilizații*". Acest lucru se va realiza folosind un screen saver special care preia date prin intermediul Internet-ului, le analizează și comunică rezultatele analizei. Dacă aveți nevoie de calculator screen saver-ul oprește imediat analiza care va continua doar după ce ați terminat lucrul.

Acest proiect este foarte interesant, dar și greu de realizat. Există atât de multe date care trebuie analizate încât pare aproape imposibil. Din fericire, analiza datelor poate fi foarte ușor împărțită în etape mai scurte care pot fi realizate simultan, în paralel. Nici una dintre aceste etape nu depinde de altele.

De asemenea, doar o porțiune finită a cerului poate fi observată de *Arecibo* (telescopul folosit pentru detectarea semnalelor extraterestre). În următorii doi ani toată porțiunea vizibilă cu ajutorul telescopului *Arecibo* va fi scanată de trei ori. Cei de la *SETI* cred că va fi suficient pentru acest proiect. Până în momentul în care și cea de-a treia cercetare a cerului va fi finalizată, vor exista noi telescoape, noi experimente și noi abordări care vor putea fi folosite pentru căutrea inteligenței extraterestre. Sperăm că veți putea participa și la aceste proiecte noi!

Unități de lucru

Datele recepționate de telescopul *Arecibo* din *Puerto Rico* sunt înregistrate pe benzi magnetice de densitate mare, în fiecare zi înregistrându-se aproximativ 35 GB de date. Deoarece *Arecibo* nu dispune de o conexiune *Internet* capabilă să realizeze transferuri de aceste dimensiuni, benzile magnetice sunt transportate până la *Universitatea Berkeley*. Datele sunt împărțite în bucăți (numite unități de lucru) având dimensiunea de 0,25 MB. Aceste unități de lucru sunt trimise de serverul *SETI@home* spre utilizatori din întreaga lume care le analizează.

SETI@home cercetează 2,5 MHz de date centrate în jurul valorii de 1.420 MHz. Acesta este încă un spectru prea larg pentru a fi trimis utilizatorilor spre analiză; pentru a soluționa și această problemă, întregul spectru este împărțit în 256 de bucăți având o lățime de bandă de 10 KHz (de fapt 9.766 Hz). Acest lucru este realizat de un produs software special denumit "*splitter*" (separator). Aceste bucăți de 10 KHz sunt mult mai ușor de manipulat. Pentru a înregistra semnale de până la 10 KHz va trebui să înregistrăm biții la o rată de 20000 bps (biți pe secundă). (Aceasta este numită *frecvența Nyquist*.) *SETI* vă va trimite aproximativ 107 secunde de înregistrări de date de 10 KHz (20 Kbps). 100 de secunde înmulțite cu cei 20.000 de biți necesari pentru înregistrarea unei secunde înseamnă 2.000.000 de biți sau aproximativ 0.25 MB. Sunt trimise și diferite alte informații suplimentare referitoare la unitatea de lucru; astfel cantitatea de informație corespunzătoare unei unități de lucru ajunge la aproximativ 340 KB.

Transferul de date

Aplicația *SETI@home* folosește legătura *Internet* numai atunci când transferă date. Acest transfer are loc doar atunci când screen saver-ul a terminat analiza unei unități de lucru și trebuie să transmită rezultatele (și să primească o nouă unitate de lucru). Conectarea la *Internet* este realizată numai cu acordul utilizatorului; acesta are posibilitatea de a controla momentul în care calculatorul său se conectează la serverul *SETI*. Există și posibilitatea de a configura aplicația

astfel încât să se conecteze la Internet imediat ce termină analiza unei unități de lucru. Potrivit afirmațiilor echipei *SETI*, transferul de date durează mai puțin de 5 minute folosind un modem obișnuit. În cadrul testelor efectuate în redacție au fost situații în care preluarea datelor corespunzătoare unei unități de lucru a durat peste 30 de minute datorită volumului tot mai mare de date pe care server-ul trebuie să le trimită utilizatorilor pentru a fi procesate.

SETI reține informații despre unitățile de lucru, folosind o bază de date uriașă. Când rezultatele analizei unei unități de lucru sunt primite de *SETI*, unitatea de lucru respectivă este marcată cu "*done*" (terminat). Se caută o nouă unitate de lucru care va fi procesată de acel utilizator și aceasta va fi marcată cu "*in progress*" (în lucru).

Știință

Să vedem acum ce realizează de fapt aplicația *SETI@home*. Să vedem mai întâi ce ne așteptăm să trimită extratereștrii. Presupunem că aceștia vor trimite un semnal folosind o modalitate cât mai eficientă care să ne permită să detectăm mesajul. Este dovedit faptul că trimiterea unui mesaj pe mai multe frecvențe nu este eficientă și necesită o putere foarte mare. Dacă mesajul este concentrat într-o gamă subțire de frecvențe (o lărgime de bandă mică) atunci mesajul este mult mai ușor de distins și nu va fi acoperit de zgomotul de fond. Acest lucru este foarte important deoarece presupunem că extratereștrii se află la o distanță foarte mare, iar semnalul lor va fi foarte slab în momentul în care va ajunge la noi. Deci nu căutăm un semnal care folosește o lărgime de bandă foarte mare (pe multe frecvențe), ci un masaj care folosește o frecvență specifică. Screen saver-ul *SETI@home* funcționează într-o manieră asemănătoare radioului atunci când se caută un anumit canal. Este cercetată puterea semnalului pentru o anumită frecvență și, în cazul în care puterea acestui semnal este apreciabilă, atunci acel semnal se va "bucura" de atenția *SETI*.

Un alt factor care ajută la eliminarea semnalelor locale (generate pe *Pământ* sau pe sateliții naturali sau artificiali ai acestuia) este acela că sursele locale sunt aproximativ constante. Astfel de semnale își vor păstra intensitatea de-a lungul timpului. Pe de altă parte telescopul *Arecibo* are o poziție fixă. Când *SETI@home* se află în acțiune, telescopul nu urmărește stelele; din acest motiv cerul "trece" prin fața focarului telescopului. Este nevoie de 12 secunde pentru ca o "țintă" să treacă de focar. De aceea, ne așteptăm ca un semnal extraterestru să devină mai puternic și mai slab timp de 12 secunde. Deoarece se caută asemenea semnale "gaussiene", *SETI* trimite aproximativ 100 de secunde de date. De asemenea, *SETI* permite datelor din unitățile de lucru să se suprapună puțin pentru ca un anumit semnal să nu fie pierdut, deoarece a fost întrerupt.

Calcule

Aplicația *SETI@home* caută semnale cu o intensitate de zece ori mai slabă decât cea a semnalelor cercetate de căutarea *SERENDIP IV* care are loc la telescopul *Arecibo*. Acest lucru este posibil deoarece *SETI@home* folosește un algoritm de calcul numit ***integrare coerentă***. Nici un alt proiect (incluzând *SERENDIP*) nu a avut la dispoziție puterea de calcul necesară pentru a implementa un astfel de algoritm. Calculatorul dumneavoastră va efectua *transformări Fourier rapide* asupra datelor primite, căutând semnale puternice pentru diferite combinații de frecvențe și lățimi de bandă și rate ale zgomotului de fond.

Vă vom prezenta acum pași care sunt efectuați pentru fiecare unitate de lucru pe care o primiți de la *SETI*. Cea mai costisitoare operație din punct de vedere a numărului de calcule este eliminarea zgomotului de fond, adică eliminarea tuturor efectelor accelerației *Doppler*. La cea mai bună rezoluție, acest lucru trebuie realizat de 5000 de ori de la -5 Hz/s la +5 Hz/s în pași de 0,002 Hz/s. Pentru fiecare rată de zgomot, se elimină zgomotul pentru cele 107 secunde de date și apoi datele sunt împărțite în 8 blocuri a câte 13,375 secunde fiecare. Fiecare bloc de 13,375 secunde este examinat pe o lățime de bandă de 0,07 Hz, căutându-se vârfuri de putere. Aceasta înseamnă 131.072 de teste (frecvențe) pentru fiecare bloc și pentru fiecare rată de zgomot. La acest prim pas calculatorul realizează aproximativ 100.000.000.000 de calcule.

Celelalte lățimi de bandă trebuie și ele studiate; la următorul pas se dublează lățimea de bandă la 0,15 Hz. Începând de la această lățime de bandă se dublează și gama ratei de zgomot (rata de zgomot va varia între -10 Hz/s și +10 Hz/s). Chiar dacă gama luată în considerare este de două ori mai mare, calculatorul va trebui să examineze doar o pătrime din numărul de rate de la pasul anterior. Se observă că la acest pas numărul de calcule care trebuie efectuate se reduce la jumătate față de primul pas, deci vom avea aproximativ 50.000.000.000 de calcule.

La următorul pas se dublează din nou lățimea de bandă (de la 0,15 la 0,3 Hz), iar numărul ratelor de zgomot se reduce la o pătrime. (Gama -10 Hz/s, +10 Hz/s va fi păstrată și pentru restul calculelor). Pentru acest pas (și pentru următorii pași) vom avea de efectuat o pătrime din numărul de calcule de la pasul anterior.

În cazul celui de-al treilea pas vom avea aproximativ 12.500.000.000 de calcule. Pentru al patrulea pas vom avea aproximativ 3.125.000.000, pentru al cincilea 781.250.000 și așa mai departe. În total vom avea 14 dublări ale lățimii de bandă (0.07, 0.15, 0.3, 0.6, 1.2, 2.5, 5, 10, 20, 40, 75, 150, 300, 600 și 1200 Hz), iar pentru cele 107 secunde de date care trebuie procesate, numărul total de calcule va depăși cu puțin 175.000.000.000. După cum puteți vedea majoritatea operațiilor sunt efectuate pentru cea mai mică lățime de bandă (aproximativ 70%).

În final, semnalele care au putere mare pentru o anumită combinație de frecvențe, lățimi de bandă și zgomote sunt testate în vederea determinării interferenței terestre. Numai dacă puterea crește și apoi descrește pe parcursul unei perioade de 12 secunde

(timpul necesar ca un punct de pe cer să treacă prin fața focarului telescopului) semnalul poate fi considerat ca având o natură extraterestră.

Potrivit celor de la *SETI*, un calculator obișnuit va avea nevoie de 10 până la 50 de ore pentru a realiza analiza unei unități de lucru. Se presupune că în acest interval de timp nu rulează nici o altă aplicație. În cazul testelor realizate de *GInfo*, pe un calculator dotat cu procesor *Intel Pentium III* având o frecvență de 500 MHz și 128 MB de memorie internă și rulând sub sistemele de operare *Microsoft Windows '98 Second Edition* și *Microsoft Windows Me* (*Millennium Edition*) analiza unei unități de lucru a durat între 7 și 8 ore. S-a observat o mică îmbunătățire a timpului necesar analizei unei unități de lucru atunci când am înlocuit *Win98se* cu *WinME*. În cazul în care în paralel cu aplicația *SETI@home* rulau și alte aplicații, analiza unei unități de lucru a durat mai mult. De exemplu, rulând și *Microsoft Word 2000* durata de procesare a unei unități de lucru a crescut cu aproximativ 20%. În cazul rulării unei aplicații care folosește intensiv procesorul (*Diablo II*) durata de procesare crește de aproximativ 10 ori.

Aplicația SETI@home

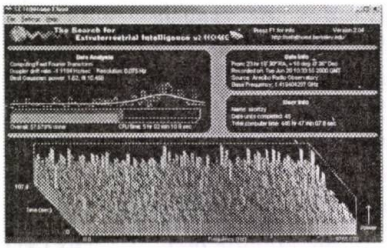
În secțiunea anterioară am spus mai multe lucruri despre aplicația propriu-zisă care este folosit pentru analiza unităților de lucru, dar mai există câteva detalii care trebuie menționate.

În primul rând, *SETI@home* poate rula și ca o aplicație obișnuită și nu doar ca un screen saver. Mai mult *SETI@home* poate rula în fundal tot timpul, dar utilizatorii trebuie să știe că astfel se consumă 15 MB de memorie. Utilizatorii care posedă calculatoare cu mai puțin de 64 MB de memorie internă nu ar trebui să folosească această facilitate.

Pentru a deveni unul dintre utilizatorii *SETI@home* va trebui să instalați aplicația și să vă înregistrați ca utilizator. Aplicația poate fi descărcată de pe site-ul *SETI@home* (http://setiathome.berkeley.edu); dimensiunea fișierului este de 771.5 KB (varianta pentru familia de sisteme de operare *Microsoft Windows*).

După instalare vă puteți înregistra; lansați în execuție aplicația, alegeți opțiunea *Create/Change Login...* și apoi apăsați butonul *Create New Account*; completați formularul și apoi apăsați butonul *Submit*; veți fi înregistrat ca utilizator *SETI@home* și veți primi prima unitate de lucru pentru procesare.

În momentul în care este deschisă aplicația *SETI@home* sau este lansat screen saver-ul corespunzător, pe ecran va apărea o prezentare grafică destul de interesantă în care se pot vedea diferite grafice.



Fereastra SETI@home

Pentru afișarea acestor grafice se consumă o mare parte din puterea de calcul a calculatorului și durata de procesare a unității de lucru crește de aproape trei ori.

Pentru a evita irosirea puterii de calcul se recomandă minimizarea ferestrei aplicației (*SETI@home* va rula în fundal și nu va mai avea nevoie să afișeze graficele) sau selectarea opțiunii *Go to blank screen* pentru screen saver. Va trebui indicat și numărul de minute care trebuie să treacă până când nu se va mai afișa nimic pe ecran.

Statistici

Probabil că cea mai interesantă parte a întregului proiect *SETI@home* constă în diferitele statistici disponibile.

Există statistici pentru fiecare țară, pentru grupuri sau pentru utilizatori individuali; toate acestea sunt disponibile pe site-ul *SETI@home*.

Criteriile de clasificare sunt foarte diverse; câteva dintre ele sunt numărul de utilizatori, numărul de unități de lucru procesate, timpul CPU folosit de calculatoare pentru procesare, timpul mediu de procesare a unei unități de lucru etc.

În data de 7 septembrie 2000 erau peste 2.300.000 de utilizatori SETI și peste 184.000.000 de unități de lucru procesate. Numărul mediu de unități de lucru procesate de un utilizator era de aproximativ 79.97. Din punct de vedere al numărului de unități de lucru procesate, în clasamentul pe țări *România* se afla pe locul 44.

Grupul GInfo

După cum am amintit anterior, utilizatorii pot forma grupuri pentru a face statisticile mai interesante.

Sperăm că sunteți interesați de acest proiect și dacă nu vă numărați încă printre utilizatorii *SETI@home*, veți deveni curând unul dintre aceștia.

Chiar dacă sunteți un utilizator nou sau unul mai vechi, sperăm că vă veți alătura grupului *GInfo*.

Pentru a vă alătura grupului nostru va trebui să intrați în pagina *Groups* a site-ului *SETI@home*, să căutați grupul *GInfo* și să apăsați pe *Join*.

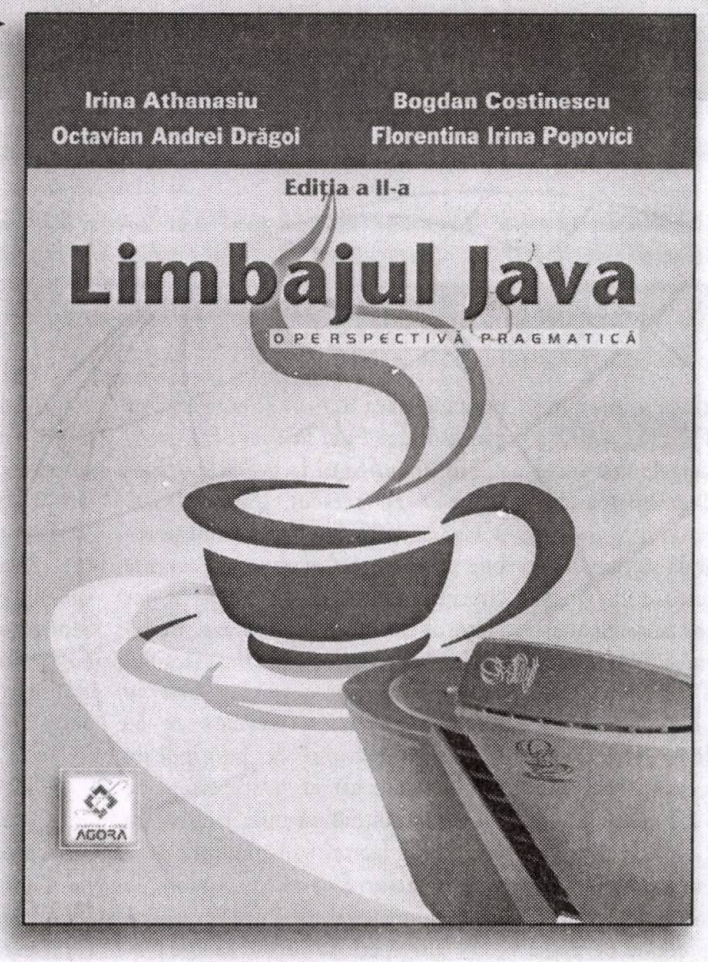
Concluzii

Nu am încercat să vă convingem că inteligența extraterestră există cu adevărat; tot ceea ce vrem să spunem este că, uneori, calculatorul dumneavoastră este în funcțiune, dar nu face decât să consume energie electrică. În acest timp puterea de calcul a calculatorului ar putea fi folosită pentru alte activități. *SETI@home* este numai unul (cel mai cunoscut) dintre proiectele care împrumută calculatoare în scopuri științifice. Deci, dacă nu vă deranjează să fiți conectați la *Internet* câteva minute pentru a primi unități de lucru, atunci calculatorul dumneavoastră ar putea fi mai folositor umanității.

SETI@home
The Search for Extraterrestrial Intelligence

Mihai Scorțaru este student în anul III la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa skortzy@email.ro.

COMPUTER LIBRIS AGORA



De ce Java ?
Deoarece tot ce se întâmplă acum în lumea calculatoarelor este pentru Internet, legat de Internet sau accesibil din Internet. Java este limbajul de programare care a devenit un standard de-facto în toate aceste cazuri. Java este acum mai mult decât un limbaj de programare, este o colecție de tehnologii pentru Internet.
De ce reeditată ?
Datorită dinamicii evoluției limbajului, autorii au considerat necesară verificarea exemplelor și pe versiunea Java 2 precum și adăugarea a două noi capitole: componente Java Beans și securitate.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

RĂZBOIUL procesoarelor (INTEL vs. AMD)

Mihai Scorțaru

Confruntarea dintre Intel și AMD rămâne în continuare cea mai interesantă competiție din domeniul IT&C. Chiar dacă în timpul verii nu au fost înregistrate creșteri importante ale vitezei de lucru, lupta a continuat prin reduceri semnificative de prețuri la procesoarele produse de cele două companii.

După ce a reușit să atingă prima barieră psihologică de 1 GHz, *AMD* a continuat să furnizeze cele mai bune procesoare destinate calculatoarelor performante.

AMD a fost capabilă să furnizeze publicului larg procesoare cu viteze de 1 GHz imediat după lansarea oficială a acestora (6 Martie 2000) în timp ce procesoarele *Intel Pentium III* cu frecvența de 1 GHz, deși au fost lansate oficial pe data de 8 Martie 2000, nu au fost disponibile pentru publicul larg decât începând cu data de 14 August 2000.

Acesta este doar un exemplu care arată că s-ar putea ca *Intel* să piardă supremația pe care o deține pe piața procesoarelor. Totuși, va mai trece o perioadă destul de lungă de timp până când *AMD* sau o altă companie va putea pretinde locul de lider în industria procesoarelor.

În cadrul acestui articol ne propunem să prezentăm situația actuală de pe piața procesoarelor, precum și tendințele care se manifestă pe această piață.

Intel Celeron

Procesoarele *Intel Celeron* sunt destinate calculatoarelor ieftine. Ultimele procesoare din această familie sunt fabricate folosind tehnologia "Coppermine", cea folosită și la fabricarea procesorelor *Intel Pentium III*.

În timpul verii frecvențele procesoarelor au crescut, atingând 700 MHz. În acest moment sunt disponibile procesoare *Celeron* cu frecvențe cuprinse între 366 și 700 MHz, iar factorul de incrementare este de 33 MHz. Acest fapt se datorează faptului că magistrala procesorelor *Celeron* funcționează la o frecvență de 66 MHz, iar factorul de multiplicare (multiplicatorul) al procesorului se obține înmulțind un număr întreg cu 0,5. Deci, factorii de multiplicare a procesoarelor *Celeron* disponibile în acest moment variază între 5,5 și 10,5.

Pentru perioada următoare se preconizează lansarea de procesoare *Celeron* de 733, 766 și 800 MHz. De asemenea, *Intel* a promis că în scurt timp vor produce și procesoare *Celeron* cu magistrala la 100 MHz. Părearea specialiștilor este că *Intel* a menținut în mod artificial magistrala procesoarelor *Celeron* la 66 MHz deoarece, folosind o magistrală de 100 MHz, performanțele procesoarelor *Celeron* ar fi comparabile cu cele ale procesoarelor *Pentium III*. Anul trecut *Intel* a introdus procesoarele *Pentium III* bazate pe tehnologia "Coppermine" care au magistrala la 133 MHz. Probabil că *Intel* va înceta producția procesoarelor *Pentium III* cu magistrala la 100 MHz în momentul în care procesoarele *Celeron* vor avea magistrala la 100 MHz.

Prețurile procesoarelor *Celeron* sunt relativ mici. O listă cu prețurile valabile în *Statele Unite* la data de 13 Septembrie 2000 este prezentată în continuare (în *România* prețurile sunt mai mari cu 10-20% și trebuie adăugat și TVA-ul):

Celeron 366	$64
Celeron 400	$70
Celeron 433	$72
Celeron 466	$67
Celeron 500	$79
Celeron 533	$80
Celeron 566	$80
Celeron 600	$84
Celeron 633	$89
Celeron 700	$168

Intel Pentium III

Procesoarele *Intel Pentium III* sunt destinate calculatoarelor performante. În timpul acestei veri nu au existat creșteri de performanță la aceste procesoare deși a existat o tentativă în acest sens.

Pe data de 31 Iulie 2000 *Intel* anunța disponibilitatea în cantități limitate a unui procesor *Pentium III* care funcționează la o frecvență de

1.13 GHz. Această lansare a fost un eșec total, procesorul fiind retras de pe piață câteva săptămâni mai târziu în urma raportării unor probleme de funcționare. Acesta este un nou semn care indică faptul că *Intel* are dificultăți în a-și păstra supremația în acest domeniu.

Un alt eveniment demn de amintit este lansarea reală a procesorului *Pentium III* cu frecvența de 1 GHz care a avut loc pe data de 14 August 2000.

În acest moment există două versiuni ale procesoarelor *Pentium III*, diferența fundamentală fiind frecvența la care funcționează magistrala: 100, respectiv 133 MHz.

Procesoarele *Intel Pentium III* cu magistrala la 100 MHz au frecvențe cuprinse între 500 și 850 MHz, factorul de incrementare fiind de 50 MHz. Există și o versiune la 450 MHz, dar aceasta nu mai este disponibilă pe piață.

În perioada următoare se preconizează încetarea producției acestui tip de procesoare, deoarece nu vor mai exista diferențe semnificative între performanțele acestora și cele ale procesoarelor *Celeron* cu magistrala la 100 MHz.

Prețurile procesoarelor *Pentium III* cu magistrala la 100 MHz sunt destul de mari și nimeni nu mai recomandă astăzi achiziționarea unui astfel de procesor. Ofertele valabile în *Statele Unite* sunt următoarele:

Pentium III 500	**$105**
Pentium III 550	**$115**
Pentium III 600	**$157**
Pentium III 650	**$159**
Pentium III 700	**$177**
Pentium III 750	**$218**
Pentium III 800	**$230**
Pentium III 850	**$322**

Procesoarele *Intel Pentium III* cu magistrala la 133 MHz au frecvențe cuprinse între 533 și 1000 MHz, cu un factor de incrementare de 66 MHz.

Intel a vrut să renunțe la acestă valoare și să-l înlocuiască cu 133 MHz, dar lansarea *Pentium III 1.13 GHz* a fost o nereușită.

În majoritatea cazurilor prețurile procesoarelor *Pentium III* cu magistrala la 133 MHz sunt puțin mai mari decât prețurile procesoarelor echivalente care au magistrala la 100 MHz. Totuși, din punct de vedere al performanței, este recomandabilă alegerea unui procesor cu magistrala la 133 MHz. Prețurile valabile la data de 13 Septembrie 2000 în *Statele Unite* sunt prezentate în continuare:

Pentium III 533	**$101**
Pentium III 600	**$158**
Pentium III 667	**$166**
Pentium III 733	**$176**
Pentium III 800	**$231**
Pentium III 866	**$341**
Pentium III 933	**$478**
Pentium III 1000	**$714**

AMD K6-2

Procesoarele *AMD K6-2* sunt destinate calculatoarelor ieftine, fiind primele procesoare care au concurat cu succes cu procesoare de la *Intel*, chiar dacă performanțele lor nu se ridică la nivelul celor ale procesorului *Celeron*.

Se pare că *AMD* va întrerupe producția acestor procesoare pentru a impune noul procesor *Duron*.

În prezent mai sunt disponibile pe piață numai cinci versiuni ale acestui procesor, funcționând la frecvențe de 450, 475, 500, 533 și 550 MHz.

În general, procesoarele K6-2 au o magistrală cu frecvența de 100 MHz, dar există și excepții. De exemplu, procesorul *AMD K6-2 475* are o magistrală cu frecvența de 95 MHz.

Prețurile procesoarelor *K6-2* sunt mici, dar din punct de vedere al performanței un K6-2 este net inferior unui *Celeron* echivalent. Lista de prețuri valabilă în *Statele Unite* la data de 13 Septembrie 2000 este următoarea:

K6-2 450	**$49**
K6-2 475	**$53**
K6-2 500	**$51**
K6-2 533	**$69**
K6-2 550	**$87**

AMD Duron

Procesorul *AMD Duron* este destinat și el calculatoarelor ieftine dar, spre deosebire de *K6-2* depășește cu mult performanțele unui *Celeron* echivalent. Din acest motiv procesorul *Duron* a fost supranumit *the Celeron killer*.

Procesorul *Duron* este fabricat folosind aceeași tehnologie ca și în cazul noii generații de procesoare *Athlon*. Magistrala funcționează la o frecvență de 100 MHz, dar sunt transmise două instrucțiuni în timpul unui tact, deci magistrala este echivalentă cu una care funcționează la o frecvență de 200 MHz.

Procesoarele *Duron* au fost lansate la frecvențe de 600, 650 și 700 MHz. Acest fapt demonstrează intenția *AMD* de a face din *Duron* un succesor al procesorului *K6-2*. 600 MHz ar fi fost următoarea frecvență logică pentru *K6-2*.

Pe data de 5 Septembrie 2000 familia procesoarelor *Duron* a primit un nou membru: *AMD Duron 750*.

Chiar dacă performanțele procesorului *Duron* sunt net superioare performanțelor procesorului *Celeron*, un *Duron* costă mult mai puțin decât un *Celeron*. Prețurile valabile în *Statele Unite* pe data de 13 Septembrie 2000 sunt următoarele:

Duron 600	**$54**
Duron 650	**$72**
Duron 700	**$91**
Duron 750	**$181**

AMD Athlon

Procesorul *AMD Athlon* este destinat calculatoarelor performante și este primul procesor care a reușit să depășească în performanță un procesor de la *Intel* care funcționa la aceeași frecvență.

După câștigarea cursei spre bariera psihologică de 1 GHz, *AMD* nu a forțat nota și nu a mai crescut frecvența procesoarelor *Athlon*. În cazul numărului vânzărilor de procesoare cu viteza de 1 GHz raportul era de 12 la 1 în favoarea companiei *AMD*. Acest fapt se datora incapacității celor de la *Intel* de a produce suficiente procesoare care să funcționeze la această frecvență. Totuși, după ce *Intel* a lansat procesorul *Pentium III* cu

frecvența de 1,13 GHz, replica celor de la *AMD* nu se putea lăsa așteptată. Astfel, pe data de 28 August 2000, era disponibil pe piață un procesor *Athlon* care funcționează la o frecvență de 1,1 GHz. Această lansare a avut loc chiar dacă *Intel* retrăsese deja de pe piață procesorul *Pentium III* cu frecvența de 1,13 GHz.

AMD a anunțat că procesoarele *Athlon* vor atinge viteza de 1,5 GHz la începutul anului următor, atunci când va trebui să concureze cu viitorul *Pentium 4* de la *Intel*. Probabil că spre sfârșitul anului 2001 procesoarele AMD vor atinge și frecvența de 2 GHz.

În acest moment există două versiuni ale procesorului *Athlon*, ambele utilizând același tip de magistrală ca și procesoarele *Duron*.

Diferența fundamentală între cele două versiuni o reprezintă viteza de lucru a memoriei *cache L2*. Inițial această memorie funcționa la doar o fracțiune din viteza procesorului (1/2, 2/5 sau 1/3), ceea ce ducea la o scădere a performanțelor. *AMD* a promis că următoarea generație de procesoare va avea o memorie *cache L2* care funcționează la aceeași viteză ca și procesorul. Numele de cod al procesorului a fost *Thunderbird* și toată lumea se aștepta ca acesta să fie lansat sub numele *Athlon II*. Spre surprinderea multora *AMD* a decis ca procesorul să se numească tot *Athlon*. Pentru a diferenția cele două tipuri de procesoare *Athlon* se folosește numele *Athlon Thunderbird* pentru noul procesor, la fel cum se folosește *Pentium III Coppermine* pentru noile procesoare *Pentium III*.

Procesorul *Athlon* "clasic" funcționează la frecvențe cuprinse între 500 și 1000 MHz, factorul de incrementare fiind de 50 Hz.

Se preconizează retragerea de pe piață a acestor procesoare deoarece procesorul *Duron* are performanțe comparabile și este mai ieftin.

Un astfel de procesor este mai ieftin decât un *Pentium III* echivalent, dar nimeni nu recomandă achiziționarea sa deoarece un *Athlon Thunderbird* este mai performant și poate fi cumpărat cu aproximativ aceeași sumă. Prețurile din *Statele Unite* sunt prezentate în continuare:

Athlon 500	**$64**
Athlon 550	**$72**
Athlon 600	**$83**
Athlon 650	**$92**
Athlon 700	**$137**
Athlon 750	**$144**
Athlon 800	**$165**
Athlon 850	**$223**
Athlon 900	**$260**
Athlon 950	**$349**
Athlon 1000	**$459**

Procesorul *Athlon Thunderbird* funcționează la frecvențe cuprinse între 700 și 1100 MHz. Până la atingerea frecvenței de 1 GHz factorul de incrementare a fost de 50 MHz, dar *AMD* a lansat *Athlon 1100* fără a mai lansa *Athlon 1050*. Probabil că în viitor factorul de incrementare va fi de 100 MHz.

Se preconizează înlocuirea magistralei de 200 MHz cu una având frecvența de 266 MHz, dar procesoare *Athlon* cu această frecvență a magistralei vor fi disponibile doar anul viitor.

Perfomanțele unui *Athlon Thunderbird* sunt superioare performanțelor unui *Pentium III*. În plus, un *Athlon Thunderbird* costă mai puțin decât un *Pentium III*. Acestea sunt principalele motive pentru care majoritatea specialiștilor consideră că *Athlon Thunderbird* este cea mai bună alegere la ora actuală. Iată și prețurile:

Athlon 700	**$134**
Athlon 750	**$150**
Athlon 800	**$172**
Athlon 850	**$228**
Athlon 900	**$266**
Athlon 950	**$349**
Athlon 1000	**$470**
Athlon 1100	**$696**

În curând...

La ora actuală aceasta este situația în industria procesoarelor. În următoarele paragrafe vom vedea la ce ne putem aștepta în următoarea perioadă de timp.

Intel Pentium 4

Următoarea generație de procesoare pe 32 de biți de la *Intel* va purta denumirea *Pentium 4*.

Acest nou procesor va fi probabil lansat în cursul lunii octombrie a acestui an.

Pentium 4 va avea magistrala de 100 MHz, dar tehnologia folosită permite transferul a patru instrucțiuni în timpul unui tact, deci magistrala este echivalentă cu una având frecvența de 400 MHz.

Intel a anunțat că, inițial, procesorul *Pentium 4* va funcționa la o frecvență de 1,4 GHz. Este destul de probabil ca în aceeași zi să fie lansată și o versiune la 1,5 GHz. Frecvențe care vor depăși 2 GHz vor fi disponibile în cursul anului următor.

Inițial *Pentium 4* va fi fabricat folosind actuala tehnologie de 0,18 microni dar, în cursul anului următor, se va trece probabil la o tehnologie de 0,13 microni.

Vor fi introduse și o serie de instrucțiuni noi care vor fi grupate într-un set numit *SSE2*.

Se pare că *Intel* nu a reușit să obțină procesorul pe care l-a dorit deoarece, potrivit majorității testelor, se pare că *Pentium 4* nu va fi cel mai performant procesor de pe piață, el nereușind să depășească cel mai bun procesor actual: *AMD Athlon 1100*. Mai mult, în unele cazuri procesorul *Pentium III* s-a dovedit mai performant decât succesorul său.

În continuare puteți vedea o prezentare a rezultelor obținute de *Intel Pentium 4 1400*, comparativ cu cele obținute de *Athlon Thunderbird 1100*.

Intel P4 - 1400 MHz

Business Winstone 99 1.2: 30.5
SysMark 2000: 180
Business Graphics 99: 337
HE Graph.99: 964
3D WB 2000: 115
3D Mark 2000: 6285
Quake 3 High Quality: 92.6
Quake 3 Normal: 194.8
Bus. Disk99: 6750
HE Disk 99: 21400
CPU Mark 32: 84
FPU: 4790

AMD Athlon - 1100 MHz
Business Winstone 99 1.2: 34.8
SysMark 2000: 206
Business Graphics 99: 513
HE Graph.99: 1406
3D WB 2000: 107
3D Mark 2000: 5764
Quake 3 High Quality: 90.9
Quake 3 Normal: 148.9
Bus. Disk99: 6450
HE Disk 99: 22300
CPU Mark 32: 97
FPU: 6010

Se observă superioritatea procesorului *Pentium 4* în cadrul testelor care verificau performanțele 3D. Totuși, pentru *Intel* ar trebui să fie îngrijorător faptul că procesorul *Athlon* a depășit procesorul *Pentium 4* la 7 dintre cele 12 teste.

Intel Itanium

Procesorul *Itanium* este primul procesor de la *Intel* care va folosi o arhitectură pe 64 de biți.

Deși ar fi trebuit să fie deja lansat, procesorul *Itanium* nu a apărut încă pe piață datorită unor dificultăți întâmpinate de *Intel*. Totuși, se pare că până la sfârșitul acestui an vom avea și un procesor pe 64 de biți.

Noua arhitectură folosită de *Intel* este total incompatibilă cu actualele aplicații pe 32 de biți, deci pentru a beneficia cu adevărat de calitățile acestui procesor vom avea nevoie de noi sisteme de operare și de noi aplicații.

Mai mulți producători de sisteme de operare au anunțat deja că vor produce sisteme de operare pe 64 de biți. Cele mai importante sisteme de operare care vor apărea sunt o versiune *Windows* de la *Microsoft*, o versiune *Solaris* de la *Sun* și o versiune *UNIX* de la *IBM/SCO*.

Pentru a putea rula aplicațiile actuale va trebui folosit un mod special de emulare care va duce la încetinirea acestora. La rularea aplicațiilor pe 32 de biți procesorul *Itanium* va fi mult mai lent decât actualul *Pentium III*.

Procesorul *Itanium* este destinat server-elor și nu va fi disponibil în cantități suficiente decât spre sfârșitul anului următor.

AMD Sledgehammer

Procesorul pe 64 de biți de la *AMD* nu a primit încă un nume oficial și este cunoscut doar sub numele de cod *Sledgehammer*.

Abordarea *AMD* diferă foarte mult de cea aleasă de *Intel*. Viitorul procesor va putea rula nativ și aplicațiile pe 32 de biți deoarece noua arhitectură va reprezenta un nou pas în evoluția actualei arhitecturi.

Data oficială a lansării procesorului pe 64 de biți de la *AMD* nu a fost încă anunțată, dar se pare că *Sledgehammer* va concura cu *Itanium* înaintea încheierii anului următor.

AMD a decis că noua arhitectură nu mai trebuie să fie compatibilă cu arhitectura folosită de *Intel*. Chiar dacă ambele arhitecturi vor rula (emulat sau nu) actualele aplicații pe 32 de biți, o aplicație pe 64 de biți scrisă pentru *Itanium* nu va putea fi rulată pe *Sledgehammer*. Acesta ar trebui să fie un motiv de îngrijorare pentru *AMD* deoarece nimeni nu a anunțat încă intenții clare de a dezvolta un sistem de operare pentru *Sledgehammer*. S-ar putea ca până la urmă să existe o versiune *Windows* și pentru *Sledgehammer*, dar până atunci *AMD* va trebui să se bazeze pe *Linux*.

Marele avantaj al firmei *AMD* este posibilitatea de a rula nativ aplicații pe 32 de biți. *Intel* spera că tranziția spre aplicațiile pe 64 de biți va fi una rapidă, dar este foarte improbabil ca acest lucru să se întâmple.

Aceasta se datorează faptului că totul depinde de existența unor compilatoare bine puse la punct. Majoritatea dezvoltatorilor de compilatoare au căzut de acord asupra faptului că dezvoltarea unor noi compilatoare pentru o arhitectură complet nouă nu mai este știință ci o artă și este nevoie de câțiva ani pentru ca aceste compilatoare să fie fiabile.

Proiectul Sledgehammer este foarte important pentru *AMD* deoarece va permite companiei să intre pe piața server-elor care este dominată clar de *Intel*.

Până atunci *AMD* intenționează să lanseze o nouă versiune *Athlon* cunoscută sub numele de cod *Mustang*. Acesta va avea o memorie *cache L2* de până la 2 MB și va concura cu procesoarele *Pentium III Xeon* de la *Intel*.

Procesoare mobile

O altă confruntare între *Intel* și *AMD* se va desfășura pe piața procesoarelor pentru notebook-uri.

La ora actuală *AMD* are o poziție destul de bună pe această piață, procesorul *K6-2+* fiind foarte apreciat. Totuși în acest sector *Intel* este regele performanței, procesoarele *Mobile Pentium III* fiind mult mai rapide decât procesoarele *Mobile K6-2+* și *Mobile K6-III+*.

AMD va lansa în curând procesorul *Mobile Athlon* și probabil va depăși rivala *Intel* și în acest domeniu.

Pe piața procesoarelor mobile viteza procesorului are un rol secundar, mult mai importante fiind posibilitățile de economisire a energiei. *Intel* va introduce tehnologia *SpeedStep* care va permite procesorului să funcționeze la o frecvență mare dacă este alimentat de la o priză și va comuta la o frecvență mai mică dacă sunt alimentate de la baterii. Tehnologia *PowerNow!* de la *AMD* este net superioară celei propuse de *Intel*. Pe lângă faptul că frecvența de lucru a procesorului se va micșora dacă este alimentat de la baterii, frecvența se va modifica și în funcție de operațiile efectuate de utilizator. Dacă utilizatorul efectuează operații simple cum ar fi citirea mesajelor *e-mail* atunci procesorul va funcționa la o frecvență redusă. Dacă utilizatorul va efectua o operație mai costisitoare cum ar fi interogarea unei baze de date mari atunci frecvența procesorului va crește.

Concluzie

Se pare că în momentul de față Intel are motive de îngrijorare. Este pentru prima dată când poziția sa de lider este amenințată de un adversar care nu reprezenta nici un pericol acum câțiva ani. Probabil că multe se vor decide în cursul anului următor.

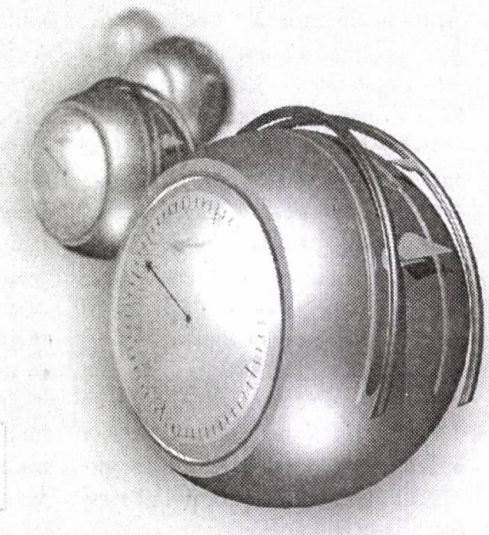
Mihai Scorțaru este student în anul III la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa skortzy@email.ro.

COMPUTER LIBRIS AGORA

DAN GRIGORAȘ

CALCULUL PARALEL

De la sisteme la programarea aplicațiilor



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate, dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performață și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Windows ME

Pe data de 14 septembrie 2000 a avut loc lansarea oficială a noului sistem de operare Microsoft Windows Me (Millenium Edition). Acesta face parte din familia Windows '9x, fiind destinat utilizatorilor de acasă. WinME introduce o serie de noi facilități, noi versiuni pentru diferite aplicații livrate împreună cu sistemul etc.

Sistemul de operare *Windows Me* părea foarte stabil încă din fazele *Beta*, motiv pentru care lansarea sa era așteptată cu nerăbdare în întreaga lume.

În *România*, *WinME* a fost lansat pe data de 18 septembrie când, în cadrul unei conferințe de presă organizate la *World Trade Center București*, a fost anunțată disponibilitatea imediată a noului sistem de operare *Microsoft* destinat utilizatorilor de acasă.

Sistemul furnizează îmbunătățiri tehnologice importante în privința modului de funcționare a PC-ului, prelucrării multimedia digitale, minirețelelor construite pentru acasă și experienței online.

Produsul poate fi cumpărat ca pachet complet sau ca upgrade pentru utilizatorii *Win95*, *Win98 sau Win98se*. Partenerii OEM din întreaga lume (printre care se află *Compaq*, *Hewlett-Packard*, *IBM*) și partenerii locali oferă *Windows Me* preinstalat pe noile PC-uri.

Windows Me înglobează cele mai noi tehnologii ale momentului pentru utilizatorii care doresc să beneficieze acasă de progresele tehnologiei informației într-un mod cât mai natural. Îmbunătățirile aduse modului de operare se concentrează pe următoarele zone:

- *Sănătatea PC-ului* - asigurându-se mijloace prin care riscurile de distrugere a informațiilor să fie evitate;
- *Prelucrări multimedia digitale* - ce permit utilizarea PC-ului pentru crearea, și gestionarea facilă a muzicii și imaginilor, asigurând interfața cu o gamă variată de dispozitive digitale (camere foto, video, scanere, playere MP3 etc.);
- Facilități pentru construirea de mini-rețele acasă;
- Crearea cadrului adecvat pentru exploatarea facilităților oferite de *Internet* (email, navigare, cumpărături online, jocuri, chat, conferințe etc.)..

Windows Me este disponibil în prezent în 14 limbi, în următoarele trei luni numărul acestora urmând să ajungă la 28. Din păcate, din cauza numărului mic de PC-uri din mediul casnic, acest sistem de operare nu va fi tradus și în limba română.

Noua versiune a sistemului de operare de la *Microsoft* permite importarea și editarea de filme prin intermediul aplicației *Windows Movie Maker*.

Folderul *My Documents* conține un subfolder numit *My Pictures*. Acesta este destinat, în principal, păstrării imaginilor digitale.

Windows Media Player este a altă aplicație folositoare. Cu ajutorul ei puteți crea și indexa librării de fișiere audio, puteți asculta posturi radio care folosesc rețeau *Internet* pentru a-și difuza programele, puteți viziona filme etc.

Windows Me nu permite ștergerea sau alterarea fișierelor sistem critice, protejându-le împotriva modificărilor sau ștergerilor accidentale, precum și împotriva virușilor.

Dacă ceva nu mai funcționează cum trebuie, aplicația *Sistem Restore* permite revenirea la o stare în care totul era OK.

Help-ul a fost și el îmbunătățit, informațiile referitoare la calculator, sistem de operare și aplicații fiind accesibile dintr-o singură locație.

Este oferit un suport îmbunătățit pentru dispozitivele *Plug&Play* actuale și viitoare.

Construirea unei rețele care să conțină un număr mic de calculatoare nu mai reprezintă o problemă. Resursele disponibile cum ar fi imprimantele vor fi detectate automat.

Outlook Express, *MSN Messenger Service* și *NetMeeting* colaborează pentru a face comunicarea mai ușoară și mai eficientă. Vă puteți conecta la *Internet* din orice colț al lumii, puteți face un chat cu prietenii sau puteți conduce o întâlnire în direct cu colegii.

Îmbunătățirea suportului pentru rețeaua de acasă duce jocurile la un nou nivel. Puteți să vă confruntați cu un prieten care se află la doi metri sau cu un adversar anonim de la 10000 de kilometri depărtare.

Conceptul de ALGORITM și importanța lui

Militon Frențiu

În ultimul timp numărul claselor de informatică din licee a crescut foarte mult, dar efectele acestui fapt nu sunt deloc benefice. Nivelul de pregătire al elevilor este din ce în ce mai scăzut. Deși au studiat patru ani informatica, mulți dintre candidații care s-au prezentat la concursul de admitere de la Universitatea Babeș-Bolyai din Cluj nu cunosc concepte fundamentale cum ar fi acela de (sub)algoritm.

În urma corectării lucrărilor scrise la proba de *Informatică*, dar și a examenelor de la disciplina *Algoritmică și Programare* cu studenții anului I, am observat că absolvenții claselor de informatică nu sunt deprinși să folosească și nu stăpânesc conceptul de subalgoritm.

Problemele date la concursul amintit sunt prezentate în continuare.

I. Să se scrie subalgoritmi pentru următoarele subprobleme:

1. suma a două polinoame;
2. produsul a două polinoame;
3. valoarea unui polinom într-un punct.

II. Să se scrie un program pentru rezolvarea următoarei probleme:

Se citesc mai multe polinoame. Un polinom se dă fie prin grad și coeficienți, fie prin monoamele sale (date prin coeficienți, exponenți) (programul va accepta ambele variante). Să se tipărească suma polinoamelor citite, produsul lor și polinomul pentru care valoarea în punctul a (dat) este maximă. Citirea se încheie când produsul polinoamelor citite are gradul mai mare decât 100, fiecare polinom citit având gradul cel mult 20.

Am formulat astfel enunțul și pentru a sugera candidaților procedurile care ar trebui folosite în programul *Pascal*, dar din 112 candidați numai 9 au înțeles acest lucru. Mai mult, marea majoritate a candidaților nu au reușit să scrie corect nici subalgoritmii ceruți la punctul I. Ei nu cunosc conceptul de ***subalgoritm***!

Dintre cei 112 candidați 64 nu au folosit deloc proceduri în programul *Pascal*, iar 39 au folosit proceduri care conțineau erori.

Nu vom analiza cauzele care au dus aici (deși ar fi interesant și acest lucru). Vom încerca să dăm soluția așteptată de la ei, ca un ghid pentru elevii actuali și pentru viitorii candidați la admiterea în facultate.

Înainte de a trece la scrierea subalgoritmilor și a programului cerut vom insista puțin asupra conceptului de subalgoritm și a erorilor grave făcute de candidați.

Când vorbim de un ***algoritm*** ne gândim și la o problemă pe care o rezolvă acest algoritm. Iar când definim un ***subalgoritm*** trebuie să explicăm cele două părți ale acestui cuvânt: *sub* și *algoritm*. În primul rând el este un algoritm pentru rezolvarea unei probleme *P*. Numai că această problemă este o parte dintr-o problemă mai complexă *C*. Iar pentru algoritmul de rezolvarea a problemei *C*, algoritmul de rezolvare a problemei *P* este un subalgoritm. De aici provine și diferența dintre algoritm și subalgoritm. În orice problemă există date cunoscute și rezultate dorite. Algoritmul problemei *C* trebuie să citească datele cunoscute în problema *C* și trebuie să tipărească rezultatele obținute. Pentru subproblema *P* datele presupuse cunoscute sunt de multe ori rezultate intermediare obținute de algoritmul problemei *C* și nu sunt cunoscute de către programator, deci nu pot fi citite! De asemenea, rezultatele obținute de subalgoritmul problemei *P* nu sunt rezultatele problemei *C* care ne interesează, ele pot fi rezultate intermediare pentru continuarea rezolvării problemei *C*. Deci nu trebuie tipărite. De aici rezultă și semnificația listei parametrilor formali din definiția unui subalgoritm și, corespunzător, a listei parametrilor actuali. Aproape toți candidații care au știut scrie subalgoritmii ceruți au citit și tipărit în subalgoritmi și aproape toți au avut subalgoritmi fără listă de parametri formali. Este obligatoriu ca această listă să conțină variabilele care marchează datele presupuse cunoscute în

subproblemă și variabilele care marchează rezultatele obținute. Este sarcina algoritmului care apelează subalgoritmul să decidă ce face cu aceste date și ce rezultate tipărește.

Cele prezentate mai sus provin și din scopul pentru care s-a inventat conceptul de algoritm. Nu este aici spațiul necesar să vorbim despre importanța subalgoritmilor în programare. Ne limităm să punctăm doar câteva avantaje:

- Reutilizabilitate: subalgoritmii se pot refolosi ori de câte ori apare subproblema respectivă în alte probleme;
- Mărirea productivității în programare și prin refolosirea subalgoritmilor existenți și prin simplificarea activității de programare. Fiind doar o parte din problema de rezolvat este mult mai ușor și mai sigur să proiectăm subalgoritmii respectivi;
- Apare posibilitatea ca în rezolvarea unei probleme să lucreze mai mulți programatori, fiecare dintre ei scriind câteva proceduri.

Pentru a putea descrie algoritmi care operează asupra polinoamelor este necesar să alegem o reprezentare a unui polinom. Evident, considerăm cunoscut conceptul de polinom și faptul că un polinom poate fi dat prin grad și coeficienți sau prin monoamele sale.

În prima variantă polinomul

$$P(X) = p_0 + p_1 \cdot X + \ldots + p_n \cdot X^n$$

este dat prin gradul n și coeficienții $(p_i, i = 0, ..., n)$, iar în a doua variantă prin perechi $(c_j, e_j), j = 1, ..., m$, perechea (c, e) reprezentând monomul $c \cdot X^e$.

Atunci când se cere valoarea unui polinom într-un punct, care se calculează cel mai ușor cu schema lui *Horner*, este avantajoasă prima reprezentare. În plus, putem conveni ca p_{-1} să păstreze valoarea lui n, deci polinomul P va fi un vector cu componentele $p_{-1}, p_0, ..., p_n$, unde n este valoarea componentei cu indicele -1. Alegem această reprezentare.

Există două modalități de a obține suma S. Pentru comparație le dăm pe amândouă, explicațiile fiind date sub formă de comentarii.

Subalgoritmul SUMAPOL1(P,Q,S) este:

```
{ S:=P+Q }
fie m:=p_-1
fie n:=q_-1
dacă m<n
  atunci s_-1:=n
  altfel s_-1:=m
sfârșit dacă     { s-a calculat gradul lui S }
pentru i:=0,m execută
  s_i:=p_i
sfârșit pentru                          { S:=P }
dacă m<n
  atunci
    pentru i:=m+1,n execută
      s_i:=0
    sfârșit pentru { s-a inițializat S }
sfârșit dacă
pentru i:=0,n execută
  s_i:=s_i+q_i
sfârșit pentru { S := S + Q = P + Q }
```

Subalgoritmul SUMAPOL2(P,Q,S) este:

```
{ S:=P+Q }
fie m:=p_-1
fie n:=q_-1;
dacă m<n
  atunci s_-1:=n
  altfel s_-1:=m
sfârșit dacă { s-a calculat gradul lui S }
dacă m<n
  atunci
    pentru i:=0,m execută
      si:=p_i+q_i
    sfârșit pentru
            { coeficienții de același grad }
    pentru i:=m+1,n execută
      s_i:=q_i
    sfârșit pentru
       { copiază coeficienții rămași din Q }
  altfel
    pentru i:=0,n execută
      s_i:=p_i+q_i
    sfârșit pentru
      { adună coeficienții de același grad }
    pentru i:=n+1;m execută
      s_i:=p_i
    sfârșit pentru
       { copiază coeficienții rămași din P }
sfârșit dacă
```

Subalgoritmul care calculează produsul a două polinoame este următorul:

Subalgoritmul PRODPOL(P,Q,R) este:

```
{ R:=P*Q }
s_-1:=m+n;          { s-a calculat gradul lui R }
pentru i:=0,m+n B
  r_i:=0
sfârșit pentru              { R:=polinomul nul }
pentru i:=0,m execută
  pentru j:=0,n execută
    r_i+j:=r_i+j+p_i*q_j
  sfârșit pentru
sfârșit pentru
```

Subalgoritmul care calculează valoarea polinomului P în punctul a, bazat pe schema lui *Horner*, este dat în continuare ca subalgoritm de tip funcție.

Funcția VALPOL(a,P) este:

```
{ VALPOL:= (a) }
fie n:=p_-1;                    {n := gradul lui P}
fie v:=p_n;
```

```
pentru i:=n-1,0 execută
  v := v*a + pi
sfârşit pentru
VALPOL:=v
```

La scrierea programului *Pascal* mai apare o procedură: cea care citeşte un polinom. Ea apare în programul *Pascal* sub numele `CITPOL(P)` şi acceptă ambele forme sub care se poate da un polinom, dar reţine în memorie un vector *P* în care `P[-1..grad]` conţine coeficienţii şi gradul polinomului. Subalgoritmul este explicat prin comentariile prezente. Scrierea programului *Pascal* este uşoară dacă este clar algoritmul folosit. Vom prezenta acest algoritm, restul explicaţiilor fiind prezente în program sub formă de comentarii.

Algoritmul `ADMITERE` este:

```
fie s_-1:=0
fie s_0:=0
  { S este polinomul sumă cerut în problemă }
fie p_-1:=0
fie p_0:=1      { P:=1; P este produsul cerut }
fie r_-1:=-1
       { R este polinomul cu valoare maximă }
repetă
  apelează CITPOL(Q)
                  { Q este polinomul citit }
  apelează SUMAPOL(S,Q,S)          { S:=S+Q }
  apelează PRODPOL(P,Q,P)          { P:=P*Q }
  fie v:=VALPOL(a,Q)
  dacă r_-1=-1
    atunci
      R:=Q
      vmax:=v    { R:= primul polinom citit }
    altfel
      dacă v>vmax atunci
        vmax:=v;
        R:=Q
          { polinomul cu valoarea mai mare }
      sfârşit dacă
  sfârşit dacă
până când p_-1>100
apelează TIPPOL(S);
apelează TIPPOL(P);
apelează TIPPOL(R)
```

Programul *Pascal* cerut este prezentat în continuare. Subliniem faptul că nu sunt reţinute în memorie polinoamele citite, aşa cum au crezut marea majoritate a candidaţilor.

Listing ADMITERE.PAS

```
Program admitere;
{solutia problemei de admitere-2000}
type polinom=array[-1..120] of Integer;
var S,P,R,Q:polinom;
    v,vmax:Integer;

procedure prodpol(P,Q:polinom;
                                  var T:polinom);
{ T:=P*Q }
var m,n,i,j:Integer;
begin
  m:=P[-1];
  n:=Q[-1];
  for i:=0 to m+n do
    T[i]:=0;  { P:=0 }
  for i:=0 to m do
    for j:=0 to n do
      T[i+j]:=T[i+j]+P[i]*Q[j]
end;

function valpol(a:Integer;
                            P:polinom):Integer;
{ valpol:=P(a) }
var i,s:Integer;
begin
  s:=0;
  for i:=P[-1] downto 0 do
    s:=s*a+P[i];
  valpol:=s
end;

procedure sumapol(P,Q:polinom;
                                  var S:polinom);
{ S:=P+Q }
var m,n,i:Integer;
begin
  m:=P[-1];
  n:=Q[-1];
  if m<n
    then S[-1]:=n
    else S[-1]:=m;
  for i:=0 to m do
    S[i]:=P[i];                          { S:=P }
  if m<S[-1] then
    for i:=m+1 to S[-1] do
      S[i]:=0;
        { initializeaza toti coeficientii }
  for i:=0 to n do
    S[i]:=S[i]+Q[i];                  { S:=S+Q }
  while (S[-1]>0) and (S[S[-1]] = 0) do
    { se elimina monoamele nesemnificative }
    S[-1]:=S[-1]-1
end;

procedure citpol(var Q: polinom);
 { citeste un polinom si-l pastreaza in Q }
 {  Q[-1]=gradul; Q[0..Q[-1]]=coeficientii }
var i,ind,coef,expo:Integer;
```

```
begin
  Write('Introduceti monoame sau grad si ');
  Write('coeficienti? (monoame=0;  ');
  Writeln('grad+coef=1)');
  Readln(ind);
  case ind of
  0: begin
       Write('Dati coeficient, ');
       Writeln('apoi exponent');
       Write('(0,0) inseamna terminarea
       Writeln(' citirii monoamelor.');
       repeat
         Readln(coef,expo);
         if q[-1]=-1 then
                    { primul monom din Q }
           begin
             Q[-1]:=expo;  { gradul }
             for i:=0 to expo-1 do
               Q[i]:=0;
             Q[expo]:=coef
           end
         else        { adauga monomul in Q }
         begin
           if expo<=Q[-1] then
                  { Gradul nu se schimba }
             Q[expo]:=coef
           else      { Gradul se schimba }
           begin
             for i:=Q[-1]+1 to expo-1 do
               Q[i]:=0;
             Q[expo]:=coef;
             Q[-1]:=expo     { noul grad }
           end
         end
       until coef=0
     end;
  1: begin
       Writeln('Dati gradul:');
       Readln(Q[-1]);
       Write('Dati coeficientii in ');
       Write('ordine crescatoare ');
       Writeln('a exponentilor:');
       for i:=0 to Q[-1] do
         Read(Q[i])
     end
  end
end;

procedure tippol(Q:polinom);
                  { Tipareste polinomul Q }
var i:Integer;
begin
  Write('Coeficientii polinomului in ');
  Write('ordine crescatoare ');
  Writeln('a exponentilor=');
  for i:=0 to Q[-1] do
    Write(Q[i]:5);
            { sau Write(Q[i]:5,'* x^',I); }
            {             ca polinom in X }
  Writeln
end;

begin
  S[-1]:=0;
  S[0]:=0;
                                    Q:=0 }
  P[-1]:=0;
  P[0]:=1;
                                  { P:=1 }
  R[-1]:=-1;
                        { R inca nu exista }
  Writeln('Introduceti punctul a:');
  Readln(a);
  repeat
    citpol(Q); sumapol(S,Q,S);    { S:=S+Q }
    prodpol(P,Q,P);               { P:=P*Q }
    v:=valpol(a,Q);
    if R[-1]=-1 then
                { R:=primul polinom citit }
      begin
        vamx:=v;
        R:=Q
      end
    else
      { R:=polinomul ca valoarea mai mare }
      begin
        vmax:=v;
        R:=Q
      end
  until P[-1]>100;
  writeln('suma polinoamelor este:');
  tippol(S);
  writeln('produsul polinoamelor este:');
  tippol(P);
  write('polinoamul cu valoarea ');
  Writeln('maxima este:');
  tippol(R);
End.
```

Dl. prof. dr. Militon Frențiu este cadru didactic la Universitatea Babeș-Bolyai din Cluj. Poate fi contactat prin e-mail la adresa mfrentiu@cs.ubbcluj.ro.

Nota redacției

Chiar dacă un elev care termină clasa a X-a ar trebui să posede cunoștințele necesare scrierii acestui program, se pare că situația reală este alta. Probabil că vina nu este numai a elevilor, ci și a profesorilor care nu acordă suficientă atenție conceptelor fundamentale în predarea informaticii. Am publicat acest articol în scopul de a trage un semnal de alarmă adresat atât elevilor cât și profesorilor lor din licee.

Așteptăm scrisorile voastre!

POȘTA redacției

Și vara aceasta am primit o mulțime de scrisori pe adresa redacției. Așa cum am promis, dialogul cu cititorii noștri se va materializa și prin intermediul acestei rubrici. Nu uitați, doar scrisorile voastre vor face posibilă existența poștei redacției în fiecare număr al GInfo!

Emilian Bold, Oțelu Roșu

Mi-a plăcut conținutul revistei și vă urez să o țineți tot așa. Aș avea totuși o plângere: prezentați prea puține lucruri. N-ar fi rău să mai adăugați niște pagini sau (dă, Doamne) un CD-ROM.

De asemenea cred că ați putea scoate news-urile ce apar pe ici și pe colo în revistă, în primul rând deoarece acestea sunt învechite în momentul în care cititorul (adică eu) le cumpără. Ca să nu mai spun de faptul că la mine-n oraș revista apare cam cu o întârziere de o lună, așa că aceste știri le-am cam aflat. Propun ca în locul lor, sau măcar printre ele, să băgați niște casete gen *Nu uita!* cu sfaturi simple pentru programare (Ex.: Inițializează-ți variabilele! *Quicksort* este de atâtea ori mai rapid decât *Bubblesort* etc.) sau să se prezinte ceva algoritmi scurți. Dacă tot găsiți ceva loc liber puteți să băgați ceva glume. [...]

N-ați putea introduce un serial *Utilizare Linux*? Toate cărțile de care-am făcut rost vorbesc despre chestii destul de simple și oricum cred *Linux* începe să prindă și pe la noi.

De asemenea mi-ar plăcea și niște articole despre grafică: generare fractali, afișare BMP-uri (de fapt asta știu), afișare GIF, JPEG...

Cu asta îmi închei scrisoarea și sper că măcar ceva din ce v-am scris să vă folosească.

P.S. N-ați putea publica și probleme mai ușoare? M-am chinuit vreo două ore să rezolv o problemă dată la națională!

Îți mulțumim pentru aprecieri! Creșterea numărului de pagini se află în discuție, dar de CD-ROM încă nu poate fi vorba. Am explicat motivele și cu alte ocazii.

Referitor la news-uri, acestea au dispărut în acest număr... Totuși, probabil ele vor reveni în noiembrie...

Serialul despre Linux va începe cât de curând posibil, articole despre fractali au mai fost în GInfo, iar cele despre diferite formate de imagini nu prea cred că interează mulți cititori. Nu are rost să scriem algoritmi pentru afișarea imaginilor atâta timp cât noile unelte de programare oferă posibilitatea afișării unei astfel de imagini prin apelarea unei simple funcții.

Glume au mai fost prin GInfo (e adevărat, cam rar), dar poate pe viitor vor fi mai multe. Totuși, a greși este omenesc, dar ca să zăpăcești de tot lucrurile e nevoie de un calculator.

Dificultatea problemelor din revistă este într-adevăr ridicată, dar rezolvarea unei astfel de probleme aduce satisfacții mult mai mari. Felicitări pentru rezolvarea problemei!

Cosmin Chiru, Constanța

Am și eu o problemă... supărătoare, oarecum. Sunt din *Constanța* și nu prea găsesc la chioșcuri revista *GInfo*. Îmi puteți spune cumva pe ce dată apare în *Constanța*? Oricum, la anul o să fac un abonament!!!

Revista GInfo este difuzată în toată țara în jurul datei de 10 a fiecărei luni în care apare. Totuși, abonamentul e mai sigur!

Liviu Lalescu, Craiova

Am participat la concursul *Bursele Agora* la problema *Rubik*. Am înțeles că problema respectivă nu poate fi rezolvată, dar totuși cred că soluția pe care am trimis-o ar putea fi considerată bună. Ideea mea de rezolvare (am găsit ceva asemănător în *Leiserson* si *Cormen*) este următoarea: calculez descompunerea matricei sistemului în matricele L și R. Aceasta parte se va rezolva în $O(n^3)$, deci acceptabil. Apoi voi considera fiecare valoare finală posibilă a matricei, pentru care voi rezolva sistemul. Având însă descompunerea L și R, acest lucru se va face în $O(kn^2)$, unde k este 1000 pentru varianta propusă la concursul *Agora*. Timpul total ar fi $O(n^3+kn^2)$.

Am verificat pentru tablouri mari (100×100) și am obținut un timp consider acceptabil (nu pentru concurs, dar ca o posibilă rezolvare) de 55 de secunde. Problema care apare este că de la dimensiuni mai mari de 35×35 apar erori mari de propagare, și programul nu mai funcționează corect. Totuși dacă s-ar implementa numere cu mai multe cifre semnificative, cred că s-ar putea rezolva problema și pentru careuri de 100×100.

Am scris cele de mai sus fiindcă nu înțeleg de ce considerați problema Rubik nerezolvabilă practic.

Nu considerăm că această problemă este nerezolvabilă. Știm doar că nimeni nu a găsit încă soluția ei. Așteptăm un articol pe această temă.

S.M.E.

Acum ai propria ta forță de reacție rapidă!

Acum poți să faci față oricăror situații neprevăzute. Acum ești înarmat cum trebuie. Noul Armada M300 cu procesor mobil Intel® Pentium® III 600 MHz cu tehnologie SpeedStep™ îți dă toată puterea de foc necesară: display de până la 12,1"; hard-disc 12 GB; numai 2,2 cm grosime; modem 56K Mini-PCI integrat. Unitatea Mobilă de Expansiune (opțional) înseamnă economie de spațiu și rază mare de acțiune, prin transformarea noului Armada M300 într-o adevărată stație de luptă: acces la încă o baterie, încă un hard disc, la CD-ROM (sau DVD), unitate de dischetă etc.
Cu noul Armada M300 de partea ta, războiul e ca și câștigat... pentru că se termină înainte de a începe.

intel inside
pentium® III

ntel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

NonStop™

net consulting

Str. Știrbei Vodă 15, București, România
Tel: 223 27 01, 223 26 80; Fax: 312 41 99
E-mail: sales@netconsulting.ro
www.netconsulting.ro

Din păcate, majoritatea consideră că o relație la birou este nepotrivită.

Iar motivul cel mai des invocat este c
ar fi contra-productivă. În realitate
nimic mai fals!
Cu noul Compaq Deskpro EN Slim
Desktop cu procesor Intel® Pentium®
866 MHz poți stabili o relație extrem
de productivă încă din prima zi. Pentru
că este un sistem compact, prietenos
și adaptabil, care nu-ți va crea
niciodată vreo problemă.
Deskpro EN Slim Desktop concentrează
tehnologie de vârf într-un spațiu c
20% mai mic decât al oricărui alt PC
hard-disc silențios Ultra ATA-100 d
până la 20GB, unitate CD-ROM, 128 M
memorie RAM, adaptor Intel Fas
Ethernet integrat și multe altele
În concluzie, o legătură de durată
inteligentă, va fi oricând acceptată d
restul colegilor ca fiind firească.
Să fie acesta singurul motiv pentru ca
ești privit cu admirație...?

intel inside
pentium® III

Intel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

NonStop™

OMNILOGIC

Șos. București - Ploiești 135
Tel: 303 31 00; Fax: 303 31 64
www.omnilogic.ro

Gifo
Noiembrie 2000
Vol. 10/7
Lei 15.000

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

PROBLEME

IOI 2000

Olimpiada Internațională de Informatică

TEHNO

intel. VS AMD

ENJOY

generația Windows

Cum să pui puterea informației în slujba ta?

Cum să hrănești mai multe gu
deodată? Și cum să le asigu
toate cele necesare fă
întrerupere?! Non stop?!
Simplu - noua generație
servere Compaq ProLiant ML5
îți sare în ajutor.
Compaq ProLiant ML530
procesor Intel® Pentium®
Xeon™ 866 MHz, cu configurat
dual-procesor este cel mai rap
server cu arhitectură 2-way d
lume. ProLiant ML530 este gând
pentru exploatarea spațiul
interior la maxim: până la 4
de memorie; 16 bay-uri; 8 slotu
PCI; capacitate maximă
depozitare a datelor: 436,8 G
Și asta nu e tot: fiabilitatea
stabilitatea sistemului su
garantate prin programe
proprietare Compaq.

intel inside
pentium® III xeon™

Intel, logotipul Intel Inside, Pentium sunt mărci înregistrate, iar Pentium III Xeon este marcă comercială a Intel Corporation.

COMPAQ

NonStop™

scop® COMPUTERS

Barbu Văcărescu 162
Tel: 231 65 78
Fax: 231 65 96
www.scop.ro

Vol. 10, Nr. 7 /2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Redactor:
Carmen Șușman (Alba Iulia)
Au mai colaborat:
Mariela Corneanu (Târgu Mureș)
Ion Cozac (Târgu Mureș)
Crina Grosan (Cluj)
Horea Haitonic (Bistrița)
Lucian Ilea (Cluj)
Iulian Pah (Cluj)
Doru Popescu Anastasiu (Slatina)
Florin Sabău (Reșița)
Ștefan Viziru (București)

Tehnoredactare:
Mihai Uricaru, Călin Borz
Coperta și grafica:
Adrian Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

probleme.. PAG. 5

propuse pentru REZOLVARE

Olimpiada Internațională de Informatică

Olimpiada Balcanică de Informatică

concurs.. PAG. 14

Enunțurile problemelor de la runda a doua

focus.. PAG. 16

GEOMETRIE analitică în C++
Horea HAITONIC

Principiul INDUCȚIEI în GEOMETRIE
Doru A. POPESCU

profil.. PAG. 20

Primii programatori
Crina GROȘAN

mate.. PAG. 27

LU-factorizare
Mihai SCORȚARU

Inegalitatea MEDIILOR și calculul RADICALILOR
Lucian ILEA
Iulian PAH

tehno.. PAG. 32

INTEL vs. AMD
Mihai SCORȚARU

PROIECTAREA bazelor de date
Ion COZAC
Mariela CORNEANU

internet.. PAG. 40

Introducere în CGI
Florin SABĂU

babel.. PAG. 45

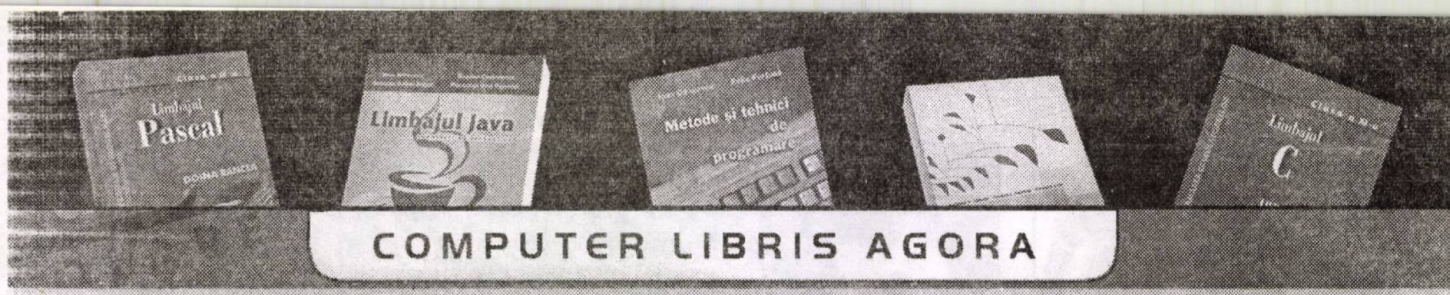
PROTECȚIA programelor
Ștefan VIZIRU

Sisteme antifurt pentru aplicații

enjoy.. PAG. 46

STAR TREK generația Windows

HOROSCOP



COMPUTER LIBRIS AGORA

"Cum aș putea să justific existența acestei cărți? Cum aș putea spre exemplu să justfic cele 100 de pagini despre programarea dinamică, acum când această metodă nu mai figurează în programele școlare? Nu știu. Sau, cum aș putea să justific capitolul despre algoritmi genetici, când pe plan mondial abia de câțiva ani a fost dovedită utilitatea și eficiența lor? Din nou nu știu! Dar rămâne bănuiala: a scrie algoritmi înseamnă a crea. Și nu poți crea fără să gândești."

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

ROMÂNIA la IOI 2000!

Cum arată un editorial scris în doi? Este prima dată când încercăm așa ceva. Probabil că nu este mai strălucită idee, dar când este nevoie, sau mai bine zis avem motive întemeiate, găsim și modalitatea de a scrie un editorial împreună. De ce îl scriem în doi? De fapt, așa cum veți vedea, îl vom scrie (aproape) în șase: patru concurenți și doi profesori, componenți ai delegației care au reprezentat *România* la cea de-a XII-a ediție a *Olimpiadei Internaționale de Informatică*, care, anul acesta, a fost organizată în *Republica Populară Chineză*, la *Beijing*. Știți ce este interesant? Conducătorii delegației am fost noi doi: ***Horia Georgescu*** (director științific al *GInfo*), și ***Clara Ionescu*** (redactor șef al *GInfo*). Concurenții sunt cititori fideli ai revistei noastre, iar doi dintre ei au participat și la concursul *Bursele Agora* de anul trecut. În plus, ***Mihai Pătrașcu*** este câștigătorul uneia dintre cele două burse.

Lăsând la o parte vraja enigmatică ce i-a cuprins pe toți cei care au aflat că anul acesta *Inter*-ul se va desfășura în *China*, țară îndepărtată, exotică, recunoaștem că am așteptat cu oarecare rețineri evenimentul. *Asia* este departe, știm atât de puțin despre poporul chinez. Totuși, această olimpiadă va intra în istoria participărilor *României* la aceste olimpiade deoarece rezultatul obținut a fost unul deosebit. Locul doi în clasamentul neoficial pe echipe a atras încă odată atenția celor prezenți asupra *României*. Profesori și concurenți se uitau la noi, unii zâmbindu-ne, alții curioși, de parcă vroiau să ne întrebe care sunt secretele succesului. Câțiva dintre ei au pus chiar întrebări, au povestit la rândul lor cu ce greutăți se confruntă, sau și-au exprimat regretul că tinerii din țara lor nu sunt dornici de performanță, mai ales dacă această performanță se obține prin muncă.

Delegații "veterani" la această olimpiadă s-au bucurat să-l revadă pe domnul profesor ***Georgescu*** în fruntea delegației noastre (nu l-au văzut de șase ani), participanții la *Olimpiada de Informatică a Europei Centrale* au continuat dialogul cu noi ca între vechi prieteni. Concurenții din echipa *României* (***Daniel Dumitran***, ***Mihai Pătrașcu***, ***Radu Ștefan*** și ***Angel Proorocu***) s-au întâlnit din nou, dornici de confruntare cu unii dintre foștii lor "adversari" de la alte concursuri din acest an (CEOI, BOI, concursul de informatică din *Yakutsk*) sau din anii precedenți.

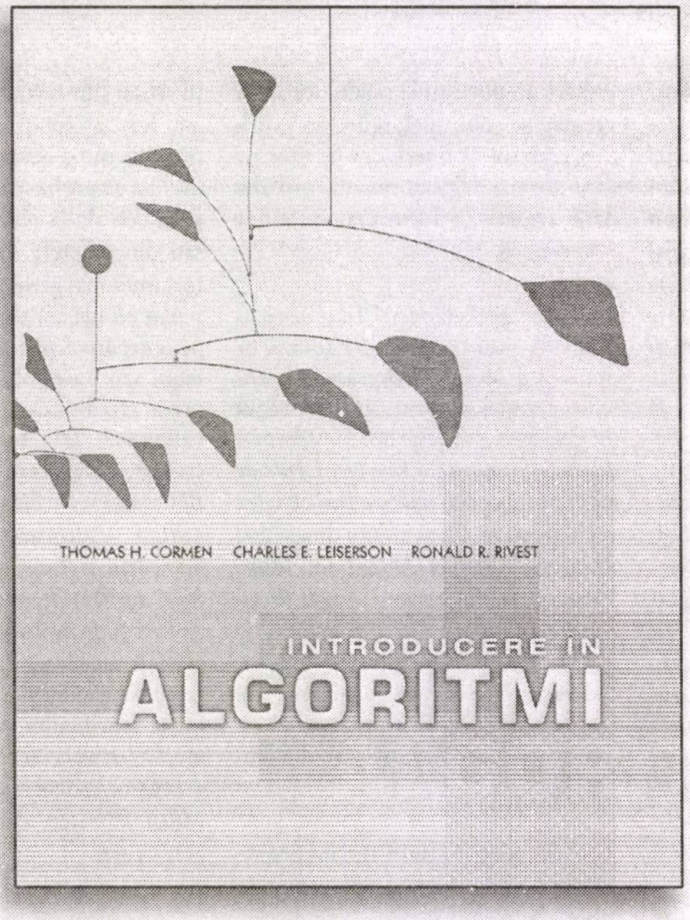
Am vrea să vă povestim despre atmosfera olimpiadei, despre emoțiile trăite, dar ele nu se deosebesc de cele trăite în alți ani, în alte țări. Olimpiada s-a desfășurat într-un centru de conferințe, mai mult european decât asiatic și, din păcate, foarte mult timp am stat între pereți obișnuiți. Organizatorii au excelat în a fi invizibili, ceea ce nu înseamnă că nu a fost o olimpiadă bine organizată, mai degrabă i-a dat o notă impersonală. Excursiile au fost interesante, dar știați că în *China* sunt foarte mulți chinezi? Și bineînțeles și ei vizitează locurile de atracție turistică ...

Am lăsat la urmă partea cu care contribuie cei patru concurenți. Am venit acasă. Am sosit la *Otopeni*, băieții și-au pus medaliile la vedere. Da, am fost așteptați de reprezentanții mass-media și acum vine surpriza. Ei declară că problemele nu au fost grele și că cele de la olimpiadele naționale sau din taberele de antrenament au fost mult mai grele. *"De ce nu am terminat olimpiada pe locul I? Pentru că ne-a depășit echipa Rusiei!"* Mici neatenții sau poate oboseala care i-a cuprins rezolvând o problemă relativ cunoscută, dar "muncitorească" (încercați să rezolvați și voi problema *Blocuri*). Pe de altă parte, ***Daniel Dumitran*** i-a surprins pe cei din *Comisia Științifică* și din *Comisia Internațională de Consiliere* a *Olimpiadei Internaționale de Informatică*, cu o soluție (la problema *Parcare auto*) la care nu s-a gândit nimeni și ca atare, datorită specificului problemei, programul lui trebuia reevaluat. Aceeași idee a avut-o, independent de ***Daniel***, și ***Mihai Pătrașcu***, dar din păcate el a greșit ceva.

Apreciem că rezultatele echipei puteau fi și mai bune, dar și băieții noștri sunt copii. Jos cu pălăria pentru problemele rezolvate cu succes, felicitări și mulțumiri profesorilor și studenților care au contribuit pe parcursul anilor la pregătirea lor. Mulțumim firmei *Romsys* care a organizat o festivitate de sărbătorire a echipei și ne-a premiat, precum și *Societății Naționale de Radiocomunicații* care, împreună cu firma *Compaq*, a apreciat că rezultatul lotului este de excepție și a acordat premii substanțiale.

Clara Ionescu, Horia Georgescu

COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

IOI 2000

Vă prezentăm enunțurile celor șase probleme cu care s-au confruntat 272 de concurenți reprezentând 72 de țări participante la cea de-a XII-a Olimpiadă Internațională de Informatică care s-a desfășurat la Beijing, în China.

P070001: Parcare auto

Un centru de parcare auto situat lângă Marele Zid este format dintr-un șir (rând) lung de locuri de parcare. O extremitate a acestui rând este considerată ca fiind cea din stânga, iar cealaltă cea din dreapta. Toate locurile de parcare sunt ocupate de automobile. Fiecare automobil este de un anumit tip, iar unele automobile din șir pot avea tipuri identice. Tipurile sunt identificate prin valori întregi. Un număr de muncitori decide să ordoneze automobilele parcate astfel încât tipurile respective să fie în ordine crescătoare de la stânga la dreapta, folosind metoda descrisă în continuare.

O rundă constă în următoarele operații: muncitorii respectivi scot simultan câte o mașină din șir, iar după această fază parchează mașinile scoase în locurile disponibile. Într-o rundă, nu este obligatoriu ca toți muncitorii să participe la operație. Pentru eficiență, este preferabil ca ordonarea să se realizeze într-un număr minim de runde.

Fie N numărul automobilelor, iar W numărul muncitorilor. Scrieți un program care, cunoscând tipul fiecărui automobil parcat și numărul muncitorilor, realizează ordonarea automobilelor într-un număr de runde care nu depășește valoarea $\lceil N/(W-1) \rceil$, unde $\lceil\ \rceil$ reprezintă partea întreagă superioară. Menționăm că numărul minim de runde nu este niciodată mai mare decât $\lceil N/(W-1) \rceil$.

Să considerăm următorul exemplu. Sunt 10 automobile parcate, având tipurile 1, 2, 3 și 4, și sunt disponibili 4 muncitori. Plasarea inițială a automobilelor redată prin tipul lor de la stânga la dreapta este:

2 3 3 4 4 2 1 1 3 1.

Numărul minim de runde este trei, rundele putând fi de exemplu, următoarele:

2 1 1 4 4 2 3 3 3 1 - după runda 1;
2 1 1 2 4 3 3 3 4 1 - după runda 2;
2 1 1 2 2 3 3 3 4 4 - după runda 3.

Date de intrare

Numele fișierului de intrare este `CAR.IN`. Prima linie conține trei numere întregi. Primul întreg este numărul N al automobilelor, $1 \le N \le 20.000$. Al doilea întreg este numărul M de tipuri, $2 \le M \le 50$. Tipurile de mașini sunt identificate prin întregi de la 1 la M. Există cel puțin câte un automobil de fiecare tip. Al treilea întreg este numărul W de muncitori, $2 \le W \le M$. Linia a doua a fișierului de intrare conține N numere întregi, unde al i-lea număr reprezintă tipul celui de-al i-lea automobil din șirul de locuri de parcare, de la stânga la dreapta.

Date de ieșire

Numele fișierului de ieșire este `CAR.OUT`. Prima linie conține un singur număr întreg R, care reprezintă numărul de runde din soluție. Următoarele R linii descriu rundele în ordine de la 1 la R. Pe fiecare linie, primul număr întreg este numărul C al automobilelor mutate în runda respectivă. Urmează $2C$ numere întregi, formând C perechi corespunzătoare deplasărilor mașinilor, utilizând pozițiile lor. Pozițiile sunt redate prin numere de la 1 la N, pornind de la capătul din stânga. Primele două numere întregi formează o pereche care descrie reamplasarea unui automobil: primul întreg reprezintă poziția de la începutul rundei, iar al doilea - poziția la sfârșitul rundei. Următorii doi întregi formează o pereche care descrie cum este mutat un alt automobil ș.a.m.d. Întrucât pot exista mai multe soluții distincte pentru aceste R linii, programul va scrie în fișierul de ieșire numai una din ele.

Exemplu

`CAR.IN`

```
10 4 4
2 3 3 4 4 2 1 1 3 1
```

`CAR.OUT`

```
3
4 2 7 3 8 7 2 8 3
3 4 9 9 6 6 4
3 1 5 5 10 10 1
```

P070002: Puterea mediană

Un experiment implică N obiecte, etichetate de la 1 la N. Se știe că N este impar. Fiecărui obiect i se atașează o in-

formație (numită putere) care este necunoscută și este exprimată printr-un număr natural; puterile obiectelor sunt distincte. Fiecare putere Y satisface relația $1 \le Y \le N$. Obiectul de putere mediană este obiectul X cu proprietatea că numărul de obiecte cu putere mai mică decât puterea lui X este egal cu numărul obiectelor cu putere mai mare decât puterea lui X. Scrieți un program care determină obiectul de putere mediană. Din nefericire, singurul mod de a compara puterile este bazat pe un dispozitiv care, pentru trei obiecte distincte date, determină pe acela dintre ele care are puterea mediană.

Biblioteci

Este dată biblioteca `device` care conține trei subprograme:

- `GetN` - trebuie apelat o singură dată, la început, nu are argumente și returnează valoarea N.
- `Med3` - trebuie apelat cu etichetele a trei obiecte distincte ca argumente și returnează eticheta obiectului de putere mediană.
- `Answer` - trebuie apelat o singură dată, la sfârșit, și are ca argument eticheta unui obiect; acest obiect ar trebui să fie cel care are puterea mediană; această operație este cea care încheie executarea programului.

Biblioteca `device` este folosită pentru a crea două fișiere text: **`MEDIAN.OUT`** și **`MEDIAN.LOG`**. Prima linie a fișierului **`MEDIAN.OUT`** conține un întreg indicând eticheta obiectului transmis ca parametru în apelarea subprogramului `Answer`. A doua linie conține un număr întreg care indică numărul de apeluri ale subprogramului `Med3` efectuate de programul dumneavoastră. Dialogul între program și bibliotecă este memorat în fișierul **`MEDIAN.LOG`**.

Programatorii în *Pascal* trebuie să includă în codul sursă instrucțiunea

```
uses device;
```

Programatorii în C/C++ trebuie să includă în codul sursă instrucțiunea

```
#include "device.h"
```

Trebuie creat proiectul **`MEDIAN.PRJ`** și apoi adăugate fișierele **`MEDIAN.C`** (**`MEDIAN.CPP`**) și **`DEVICE.OBJ`** în acest proiect.

Experimentare

Lucrul cu biblioteca se poate experimenta creând un fișier text **`DEVICE.IN`**. Fișierul trebuie să conțină două linii. Prima linie conține un întreg: numărul N al obiectelor. A doua linie conține numerele întregi de la 1 la N într-o ordine oarecare, al i-lea întreg reprezentând puterea obiectului etichetat cu i.

Exemplu

`DEVICE.IN`

```
5
2 5 4 3 1
```

Fișierul descrie o intrare cu cinci obiecte și puterile lor:

Etichetă:	1	2	3	4	5
Putere:	2	5	4	3	1

O secvență corectă este formată din următoarele cinci apeluri:

1. `GetN` (în *Pascal*) sau `GetN()` (în *C/C++*) - returnează `5`.
2. `Med3(1,2,3)` - returnează `3`.
3. `Med3(3,4,1)` - returnează `4`.
4. `Med3(4,2,5)` - returnează `4`.
5. `Answer(4)`

Restricții

- Numărul N al obiectelor este impar și $5 \le N \le 1499$.
- Orice etichetă i satisface relația $1 \le i \le N$.
- Orice putere Y satisface relația $1 \le Y \le N$; puterile sunt distincte.
- Numele bibliotecii *Pascal* este **`DEVICE.TPU`**.
- Declarațiile subprogramelor *Pascal* sunt

```
function GetN:Integer;
function Med3(x,y,z:Integer):Integer;
procedure Answer(m:Integer);
```

- Numele bibliotecilor *C/C++*: **`DEVICE.H`**, **`DEVICE.OBJ`** (folosiți modelul de memorie *large*).
- Header-ele funcțiilor *C/C++* sunt:

```
int GetN(void);
int Med3(int x,int y,int z);
void Answer(int m);
```

- La o rulare sunt permise cel mult 7777 de apeluri ale subprogramului `Med3`.
- Programul dumneavoastră nu trebuie să conțină instrucțiuni explicite de citire/scriere din/în vreun fișier; citirile/scrierile se realizează prin intermediul subprogramelor din biblioteci.

P070003: Palindrom

Un palindrom este un șir simetric de caractere, adică un șir care, citit atât de la stânga la dreapta cât și de la dreapta la stânga, este identic. Scrieți un program care, pentru un șir dat, determină numărul minim de caractere care trebuie inserate în el, inclusiv la capete, pentru a obține un palindrom. De exemplu, inserând două caractere, șirul `Ab3bd` poate fi transformat într-un palindrom (`dAb3bAd` sau `Adb3bdA`). Se observă că inserarea a mai puțin de două caractere nu duce la obținerea unui palindrom.

Date de intrare

Fișierul de intrare este **`PALIN.IN`**. Prima linie conține un număr întreg: lungimea N a șirului, $3 \le N \le 5000$. A doua linie conține un șir de lungime N. Șirul este format din literele mari de la `'A'` la `'Z'`, literele mici de la `'a'` la `'z'` și cifrele de la `'0'` la `'9'`. Literele mari și mici sunt considerate distincte.

Date de ieșire

Fișierul de ieșire este **`PALIN.OUT`**. Acesta trebuie să conțină un singur număr întreg, care reprezintă minimul căutat.

Exemplu

PALIN.IN

```
5
Ab3bd
```

PALIN.OUT

```
2
```

P070004: Descompunere în blocuri

Un cub unitate este un cub de dimensiuni $1 \times 1 \times 1$; coordonatele x, y, z ale colțurilor sale sunt numere întregi. Două cuburi unitate se consideră alipite dacă au o față comună. Prin obiect tridimensional înțelegem o mulțime nevidă de cuburi unitate alipite două câte două (vezi figura 1). Volumul obiectului este numărul cuburilor unitate din care este format.

Fig. 1: Cal

Un bloc este un obiect cu volumul cel mult egal cu 4. Două blocuri sunt de același tip dacă pot fi obținute unul din altul prin translații și rotații (dar nu prin oglindire). Există exact 12 tipuri de blocuri (vezi figura 2). Culorile din figură nu prezintă importanță; ele sunt folosite doar pentru a prezenta mai clar structura obiectelor.

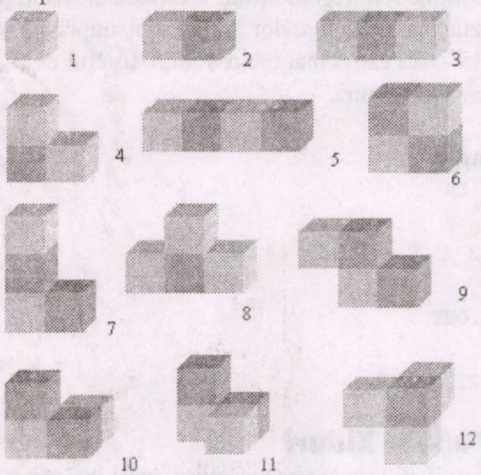


Fig. 2: Cele 12 tipuri de blocuri

Printr-o descompunere a unui obiect S se înțelege o mulțime D de blocuri din care poate fi reconstituit obiectul S. Precizăm că două blocuri din D nu pot avea un cub unitate în comun.

Scrieți un program care, pentru o descriere a tipurilor de blocuri și un obiect S, determină acea descompunere a lui S care cuprinde un număr minim de blocuri având tipurile date. Trebuie determinate doar tipurile blocurilor, așa cum apar în descompunere.

Date de intrare

În fișierul de intrare un cub unitate este descris prin trei numere întregi x, y, z, reprezentând coordonatele acelui colț pentru care suma $x + y + z$ este minimă.

Fișierul de intrare TYPES.IN conține descrierea celor 12 tipuri de blocuri din figura 2, în ordinea numerelor atașate tipurilor blocurilor. Acest fișier este folosit pentru toate testele. Fiecare bloc este descris printr-o secvență de linii consecutive. Prima linie conține numărul I al tipului de bloc ($1 \le I \le 12$). A doua linie conține volumul blocului ($1 \le V \le 4$). Următoarele V linii conțin cuburile unitate componente, prin precizarea întregilor x, y, z, care le descriu ($1 \le x, y, z \le 4$).

Fișierul BLOCK.IN conține descrierea obiectului care trebuie descompus. Prima linie conține volumul V al obiectului ($1 \le V \le 50$). Următoarele V linii conțin, fiecare, un triplet de numere întregi x, y, z ce descrie câte un cub unitate din componența obiectului ($1 \le x, y, z \le 7$).

Date de ieșire

Fișierul de ieșire este BLOCK.OUT. Prima linie va conține întregul M, reprezentând numărul minim de blocuri în care obiectul descris în BLOCK.IN poate fi descompus.

A doua linie va conține M numere, reprezentând numerele atașate tipurilor de blocuri în care obiectul a fost descompus.

Dacă există mai multe soluții, în fișier se va scrie una singură.

Eexemplu

TYPES.IN

```
1
1
1 1 1
2
2
1 1 1
1 2 1
3
3
1 1 1
1 2 1
1 3 1
4
3
1 1 1
1 2 1
1 3 1
1 4 1
6
4
1 1 1
1 2 1
1 1 2
1 2 2
```

BLOCKS.IN

```
18
2 1 1
4 1 1
2 3 1
4 3 1
2 1 2
3 1 2
4 1 2
1 2 2
2 2 2
3 2 2
4 2 2
2 3 2
3 3 2
4 3 2
4 2 3
4 2 4
4 2 5
5 2 5
```

BLOCK.OUT

```
5
7 10 2 10 12
```

```
7
4
1 1 1
1 2 1
1 1 2
1 1 3
8
4
1 1 1
1 2 1
1 3 1
1 2 2
9
4
1 2 1
1 3 1
1 1 2
1 2 2
10
4
2 1 1
1 2 1
2 2 1
2 1 2
11
4
1 1 1
1 2 1
2 2 1
1 1 2
12
4
2 2 1
2 1 2
1 2 2
2 2 2
```

Note

1. Fișierul **BLOCK.IN** corespunde "calului" din figura 1.
2. Alte variante pentru cea de-a doua linie din fișierul de ieșire sunt:

```
2 7 10 11 12
2 7 11 11 12
4 4  7 10 11
4 4  9 10 11
```

P070005: Oficii poștale

Se consideră o autostradă dreaptă cu orașe situate de-a lungul ei. Autostrada reprezintă o axă având numai coordonate întregi, deci poziția fiecărui oraș este identificată printr-un număr întreg. Nu există două orașe situate pe aceeași poziție. Distanța între două poziții este valoarea absolută a diferenței coordonatelor lor.

În unele orașe, dar nu neapărat în toate, urmează a fi amplasate oficii poștale. Oficiile poștale pot fi situate doar în orașe, iar fiecare oraș poate avea cel mult un oficiu poștal. Dându-se numărul de oficii poștale, se caută o amplasare a lor astfel încât suma tuturor distanțelor între fiecare oraș și oficiul poștal cel mai apropiat de acesta să fie minimă.

Scrieți un program care, dându-se pozițiile orașelor și numărul de oficii poștale, determină atât cea mai mică sumă posibilă a distanțelor de la fiecare oraș la oficiul poștal cel mai apropiat de acesta, cât și pozițiile unde sunt amplasate oficiile poștale astfel încât să se obțină această sumă minimă.

Date de intrare

Fișierul de intrare este **POST.IN**. Prima linie conține două numere întregi: primul reprezintă numărul V al orașelor, $1 \le V \le 300$, iar al doilea reprezintă numărul P al oficiilor poștale, $1 \le P \le 30$ și $P \le V$. A doua linie conține V numere întregi în ordine crescătoare, reprezentând pozițiile orașelor. Pentru fiecare poziție X avem $1 \le X \le 10000$.

Date de ieșire

Fișierul de ieșire este **POST.OUT**. Prima linie conține un întreg S, reprezentând cea mai mică sumă posibilă a distanțelor de la fiecare oraș la oficiul poștal cel mai apropiat de acesta, conform descrierii din cea de a doua linie. A doua linie conține P întregi în ordine crescătoare. Acești întregi reprezintă pozițiile orașelor în care sunt amplasate oficiile poștale. Dacă există mai multe soluții, fișierul de ieșire va conține una singură.

Exemplu

POST.IN

```
10 5
1 2 3 6 7 9 11 22 44 50
```

POST.OUT

```
9
2 7 22 44 50
```

P070005: Ziduri

Într-o țară au fost construite ziduri astfel încât fiecare zid leagă două orașe. Zidurile nu se intersectează și împart țara în zone închise. Deplasarea dintr-o zonă în alta se poate face trecând dintr-un oraș într-un oraș vecin sau traversând ziduri. Între oricare două orașe A și B există cel mult un zid, având drept capete pe A și pe B. Rețeaua de ziduri este conexă.

În țară există un club ai cărui membrii locuiesc în orașe diferite. În fiecare oraș locuiește cel mult un membru al clubului.

Deseori membrii clubului doresc să se întâlnească undeva în afara orașelor, în una dintre zone. Membrii clubului călătoresc cu biciclete. Din cauza traficului intens ei nu vor să treacă prin orașe și astfel sunt nevoiți să traverseze ziduri. Traseul unui membru către o zonă nu va trece prin nic un oraș cu excepția celui de plecare. Evident, numărul de traversări de ziduri trebuie să fie cât mai mic.

Pentru a ajunge în zona de întâlnire, fiecare membru traversează un număr de ziduri, eventual nici unul. Ei doresc să stabilească o zonă (numită optimă) de întâlnire, pentru care suma acestor numere, numită sumă de traversare, este minimă.

Orașele sunt etichetate cu numere întregi de la 1 la N, unde N este numărul de orașe. În figura 1 vârfurile reprezintă orașe, iar liniile care le leagă reprezintă ziduri. Presupunem că cei trei membri ai clubului locuiesc în orașele 3, 6 și 9.

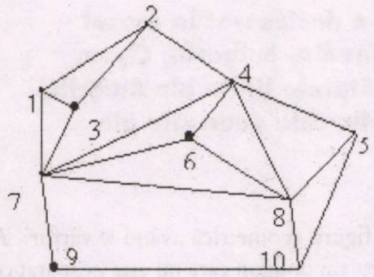


Figura 1

Zona optimă și traseele respective pentru fiecare membru sunt indicate în figura 2. Suma de traversare minimă este 2: membrul din orașul 9 traversează zidul dintre orașele 2 și 4, iar cel din orașul 6 zidul dintre orașele 4 și 7.

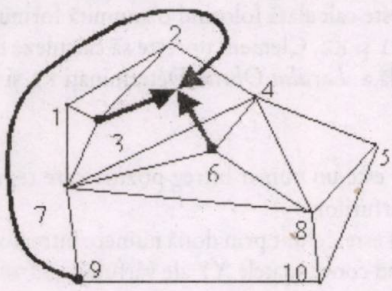


Figura 2

Scrieți un program care, pornind de la descrierea orașelor, a zonelor și a orașelor în care locuiesc membrii clubului, determină zona optimă și suma de traversare minimă.

Date de intrare

Numele fișierului de intrare este **WALLS.IN**. Prima linie conține un întreg: numărul M al zonelor, $2 \le M \le 200$. Linia a doua conține un întreg: numărul N al orașelor, $3 \le N \le 250$. Linia a treia conține un întreg: numărul L ai membrilor clubului, $1 \le L \le 30$, $L \le N$. Linia a patra conține, în ordine crescătoare, L numere întregi distincte: etichetele orașelor în care locuiesc membrii clubului.

În continuare fișierul de intrare conține $2M$ linii, câte o pereche de linii pentru fiecare zonă: primele două dintre cele $2M$ linii descriu prima zonă, următoarele două linii descriu a doua zonă ș.a.m.d. Într-o pereche de linii, prima conține numărul I al orașelor situate pe frontiera zonei respective. A doua linie din pereche conține cele I etichete ale acestor orașe, scrise în ordinea parcurgerii frontierei zonei respective în sensul acelor de ceasornic.

Ultima zonă din fișierul de intrare este aceea care împrejmuiește toate orașele și toate celelalte zone; de această dată ordinea etichetelor este specificată în sensul invers acelor de ceasornic. Ordinea zonelor din fișierul de intrare determina etichetarea acestora: prima zonă are eticheta 1, zona a doua are eticheta 2 ș.a.m.d.

Accentuăm faptul că fișierul de intrare conține toate zonele formate de orașe și ziduri, precum și zona formată de zidurile externe.

Date de ieșire

Numele fișierului de ieșire este **WALLS.OUT**. Prima linie conține un întreg: suma de traversare minimă. Linia a doua conține un întreg: eticheta zonei optime. Întrucât pot fi mai multe soluții distincte, programul va determina numai una dintre ele.

Exemplu

WALLS.IN

```
10
10
3
3 6 9
3
1 2 3
3
1 3 7
4
2 4 7 3
3
4 6 7
3
4 8 6
3
6 8 7
3
4 5 8
4
7 8 10 9
3
5 10 8
7
7 9 10 5 4 2 1
```

WALLS.OUT

```
2
3
```

BOI 2000

Olimpiada Balcanică de Informatică din anul 2000 s-a desfășurat în orașul Ohrid din Macedonia. Au participat concurenți din Albania, Bulgaria, Cipru, Georgia, Grecia, Iugoslavia, Macedonia, România și Statele Unite ale Americii. În continuare vă prezentăm enunțurile problemelor din cele două zile ale competiției.

P070007: Piața

Pescarul Clement a prins N pești și i-a adus la piața de pește. Angajații de la piața de pește împachetează peștii în pachete cu diferite greutăți. Fiecare pachet poate avea unul sau mai mulți pești.

Să presupunem că sunteți primul client și puteți alege greutatea pachetului dumneavoastră. Câte greutăți diferite putem obține pentru pachete?

Limite

- $N \leq 500$ este un număr întreg pozitiv care indică numărul peștilor.
- $t_i \leq 1000$ este un număr întreg pozitiv care reprezintă greutatea celui de-al i-lea pește.

Timp de execuție: 75 de secunde

Fișier de intrare: MARKET.IN

Pe prima linie a fișierului de intrare se află numărul întreg pozitiv N. Pe fiecare dintre următoarele N linii se află un număr întreg pozitiv t_i.

Fișier de ieșire: MARKET.OUT

Ieșirea constă într-un număr pozitiv întreg K care reprezintă numărul greutăților distincte care pot fi obținute.

Exemplu

MARKET.IN	MARKET.OUT
5	27
800	
200	
354	
18	
182	

P070008: Dimensiune fractală

Clement a învățat la școală cum să calculeze dimensiunea fractală a unei figuri geometrice. În acest scop el desenează o grilă unitate având coordonate întregi. Apoi desenează pe grilă o figură geometrică având N vârfuri. Aceste N vârfuri definesc un poligon care nu este neapărat convex și ale cărui laturi nu se intersectează. El trebuie să numere cele K1 pătrate unitate de pe grilă care intersectează figura. Prin *intersecție* înțelegem faptul că pătratul are cel puțin un punct comun cu interiorul figurii. După aceea șterge liniile grilei care au coordonate impare și numără cele K2 pătrate ale noii grile care intersectează figura. Dimensiunea fractală este calculată folosind o anumită formulă care depinde de K1 și K2. Clement dorește să calculeze dimensiunea fractală a *Lacului Ohrid*. Determinați K1 și K2.

Limite

- $2 < N < 1000$ este un număr întreg pozitiv care reprezintă numărul vârfurilor.
- Fiecare vârf este definit prin două numere întregi x_i și y_i reprezentând coordonatele XY ale vârfului($-32000 \leq x_i, y_i \leq 32000$).
- fiecare vârf este legat de vârful precedent printr-o linie dreaptă; primul vârf este și el legat de ultimul vârf.

Timp de execuție: 1 secundă

Fișier de intrare: FRACTAL.IN

Prima linie a fișierului de intrare conține numărul întreg N. Fiecare dintre următoarele N linii conține două numere întregi x_i și y_i.

Fișier de ieșire: FRACTAL.OUT

Ieșirea constă într-o singură linie care conține numerele K1 and K2.

Exemplu

FRACTAL.IN	FRACTAL.OUT
4	110 30
0 0	
10 0	
20 10	
10 10	

P070009: Scufundări

În fiecare an în *Ohrid* se desfășoară cursuri de scufundări. În cadrul acestor cursuri se desfășoară diferite activități. Una dintre ele este scufundarea într-o parte a unei stânci, trecerea printr-un pasaj subacvatic și ieșirea de cealaltă parte a stâncii. Întregul grup de N studenți dispune de un singur tub de oxigen. Maximum două persoane pot folosi tubul de oxigen în același timp. Fiecare scufundător are o viteză proprie și atunci când doi studenți se scufundă împreună, ei se vor deplasa cu viteza celui mai lent dintre ei. Există M perechi de studenți care nu se plac unul pe altul și care nu se pot scufunda în același timp.

Stabiliți un program pentru a muta cât mai repede grupul de studenți dintr-o parte a stâncii în cealaltă parte.

Limite

- $N \leq 6000$ este un număr întreg pozitiv care reprezintă numărul studenților din grup. Studenții sunt etichetați cu numerele 1, 2, ..., N.
- $M \leq 6000$ este un număr întreg pozitiv care reprezintă numărul perechilor de studenți care nu se pot scufunda împreună.
- t_i este un număr întreg pozitiv care reprezintă timpul necesar celui de-al i-lea pentru a trece dintr-o parte în alta a stâncii.
- Tubul de oxigen poate fi mutat dintr-o parte în alta a stâncii pe sub apă. Tubul de oxigen are capacitatea nelimitată. Există întotdeauna cel puțin o soluție.

Timp de execuție: 1 secundă

Fișier de intrare: DIVING.IN

Prima linie a fișierului de intrare conține două numere întregi N și M. Fiecare dintre următoarele N linii conține un număr întreg t_i. Fiecare dintre următoarele M linii conține două numere întregi reprezentând o pereche de studenți care nu se pot scufunda împreună.

Fișier de ieșire: DIVING.OUT

Prima linie a fișierului de ieșire va conține un număr întreg reprezentând timpul minim necesar pentru traversarea tuturor studenților. Următoarele linii descriu un program de traversare în timp minim. Fiecare linie reprezintă o traversare și constă din unul sau două numere întregi reprezentând studentul (studenții) care participă la traversarea respectivă.

Exemplu

DIVING.IN	DIVING.OUT
4 2	6
1	3 1
2	1
1	4 2
2	3
3 4	3 1
2 3	

P070010: Poduri

Naum s-a întors în satul său *Elshani* cu o grămadă de bani. În fiecare zi locuitorii din *Elshani* se duc la muncă în unul dintre satele vecine *Peshtani* și *Lagadin*. Există o mulțime de drumuri prin munți până la destinația lor. Pentru a-ți ajuta vecinii să-și călătoriile *Naum* dorește să construiască un pod care poate avea lungimea maximă Lmax.

Găsiți locul în care poate fi construit un pod care satisface următoarele criterii:

- Va exista un drum nou de la *Elshani* la *Peshtani* mai scurt decât cel mai scurt drum existent
- Va exista un drum nou de la *Elshani* la *Lagadin* mai scurt decât cel mai scurt drum existent
- Suma lungimilor noilor drumuri minime va fi minimă.

Limite

- $3 \leq N \leq 2000$ este un număr întreg pozitiv reprezentând numărul de puncte.
- Punctele sunt etichetate cu 1, 2, ..., N, unde 1 reprezintă punctul de pornire (satul *Elshani*), 2 este una dintre destinații (satul *Peshtani*) și 3 este cealaltă destinație (satul *Lagadin*). Două puncte pot fi conectate printr-un singur punct. Lungimea unui drum este cea mai scurtă distanță dintre punctele corespunzătoare. Din fiecare punct pleacă cel mult 20 de drumuri.
- $-32000 \leq x_i, y_i \leq 32000$ sunt numere întregi reprezentând coordonatele XY ale celui de-a i-lea punct.
- $M \leq 5000$ este un număr întreg pozitiv reprezentând numărul drumurilor dintre puncte.
- Lmax este un număr real pozitiv reprezentând lungimea maximă a podului.
- Erorile de rotunjire nu vor afecta rezultatul obținut.
- Dacă există o soluție atunci aceasta este unică.

Timp de execuție: 15 secunde

Fișier de intrare: BRIDGE.IN

Prima linie a fișierului de intrare conține două numere întregi N și M și un număr real Lmax. Fiecare dintre următoarele N linii conține două numere x_i și y_i. Fiecare dintre următoarele M linii conține două numere întrgi reprezentând punctele legate de un drum.

Fișier de ieșire: BRIDGE.OUT

Dacă există o soluție, ieșirea constă în două numere întregi K_1 și K_2 reprezentând punctele între care va fi construit podul. Dacă podul nu poate fi construit în condițiile date, se va scrie 0 0 pe prima linie, iar pe a doua linie se vor scrie două numere reale d_1 și d_2 cu o precizie de 0 (zero) zecimale. d_1 este drumul minim între punctele 1 și 2, iar d_2 este drumul minim între punctele 1 și 3.

P070011: Păianjen

Un păianjen și un țânțar se află într-o cutie de sticlă de dimensiuni L, M și N. Ei se deplasează pe suprafețele cutiei. Ei nu pot zbura și nu cad, indiferent de suprafața pe care

se află. Găsiți o strategie pentru ca păianjenul să prindă țânțarul cât mai repede.

Programul dumneavoastră va trebui să joace împotriva unor module controlate de calculator. Cei doi jucători mută alternativ. Programul dumneavoastră va controla mișcările păianjenului, iar modulul controlat de calculator pe cele ale țânțarului.

Limite

- `L,M,N≤10000` sunt numere întregi reprezentând dimensiunile cutiei.
- Cutia se află într-un sistem de coordonate *XYZ* cu un colț în origine și celelalte colțuri pe axe.
- Păianjenul și țânțarul se pot afla numai în puncte de coordonate întregi pe suprafața cutiei, după cum se arată în figura 1.

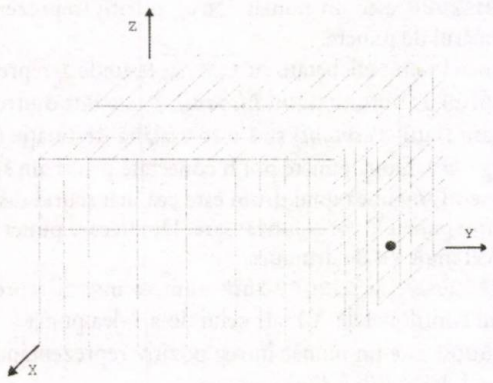


Figura 1: Suprafețele cutiei

- La o mutare păianjenul poate sări așa cum se arată în figura 2.

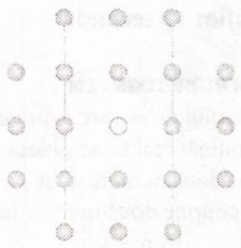


Figura 2: Mutările păianjenului

- La o mutare țânțarul poate sări așa cum se arată în figura 3.

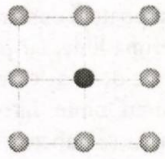


Figura 3: Mutările țânțarului

- Atât păianjenul cât și țânțarul se pot muta de pe o suprafață pe alta trecând peste o muchie.
- Păianjenul prinde țânțarul dacă reușește să ajungă în aceeași poziție cu acesta.

Timp de execuție: 4 secunde

Fișier de intrare: `SPIDER.IN`

Prima linie a intrării constă din trei numere întregi `L`, `M` și `N`. Cea de-a doua linie constă din trei numere întregi x_s, y_s, z_s reprezentând poziția inițială a păianjenului. Ultima linie constă din trei numere întregi x_b, y_b, z_b reprezentând poziția inițială a țânțarului.

Fișier de ieșire: `SPIDER.OUT`

Apelul subprogramului `Finish` produce un fișier de ieșire. Acest fișier nu poate fi modificat de programul dumneavoastră. Fișierul va conține numărul de mutări, indicația că programul a funcționat corect sau motivul pentru care acesta nu a funcționat corect.

Evaluare

Un test nu este punctat dacă apar:

- Două apeluri consecutive pentru `movespider`.
- Două apeluri consecutive pentru `movebug`.
- Mutări nepermise ale păianjenului.

Fie `T` numărul minim de mutări ale păianjenului necesare pentru a prinde țânțarul și `S` numărul de mutări ale păianjenului efectuate de programul dumneavoastră.

Dacă `S<T+3` atunci testul va fi punctat cu 10 puncte. Dacă `T+3≤S<1.2*T` atunci testul va fi punctat cu 8 puncte. Dacă `1.2*T≤S<1.4*T` atunci testul va fi punctat cu 5 puncte.

Instrucțiuni pentru programatorii în Pascal

Veți avea un unit numit `USpider.tpu`. Sursa dumneavoastră va trebui să conțină la început instrucțiunea:

```
uses USpider;
```

Unit-ul conține 4 proceduri: `init`, `movespider`, `movebug` and `finish`.

```
procedure Init;
```

Procedura `Init` ar trebui apelată la începutul programului.

```
procedure movespider(xns,yns,zns:Integer);
```

Procedura `movespider` este folosită pentru a indica noile coordonate ale păianjenului după o mutare.

```
procedure movebug(var xnb,ynb,znb:Integer);
```

Procedura `movebug` implementează strategia calculatorului și returnează noile coordonate ale țânțarului după o mutare.

```
procedure Finish;
```

Procedura `Finish` trebuie apelată după ce păianjenul a prins țânțarul. Această procedură va întrerupe și execuția programului.

La început programul dumneavoastră va trebui să apeleze procedura `Init`. Apoi programul va trebui să apeleze alternativ procedurile `movespider` și `movebug`. După apelul unei proceduri `movespider`, care mută păianjenul în poziția țânțarului, va trebui apelată procedura `Finish`.

Instrucțiuni pentru programatorii în C/C++

Veți avea un fișier obiect numit `USpider.obj` și un fișier header numit `USpider.h`.

Înaintea rezolvării problemei va trebui să deschideți un nou proiect cu numele `Spider.prj` și care va conține două fișiere: `USpider.obj` și fișierul care conține codul sursă. La începutul fișierului sursă va trebui să folosiți instrucțiunea: `#include "USpider.h"` sau instrucțiunea `#include <USpider.h>`.

Fișierul `Spider.obj` conține 4 funcții: `Init()`, `movespider()`, `movebug()` și `Finish()`.

```
void Init();
```

Funcția `Init()` trebuie apelată la începutul programului.

```
void movespider(int xns,int yns,int zns);
```

Funcția `movespider()` este folosită pentru a indica noile coordonate ale păianjenului după o mutare.

```
void movebug(int *xnb,int *ynb,int *znb);
```

Funcția `movebug()` implementează strategia calculatorului și returnează noile coordonate ale țânțarului după o mutare.

```
void Finish();
```

Funcția `Finish()` trebuie apelată după ce păianjenul prinde țânțarul. Această funcție întrerupe și execuția programului.

La început programul dumneavoastră va trebui să apeleze funcția `Init()`. Apoi programul va trebui să apeleze alternativ funcțiile `movespider()` și `movebug()`. După apelul unei proceduri `movespider()`, care mută păianjenul în poziția țânțarului, trebuie apelată funcția `Finish()`.

Exemplu

SPIDER.IN

```
10 12 8
10 11 1
5 12 2
```

Interfața cu unitul	*Coord. returnate*
`Init;`	
`movespider(9,12,2);`	
`movebug(xb, yb, zb) ;`	(4, 12, 1)
`movespider(7,12,1);`	
`movebug(xb, yb, zb);`	(3, 12, 0)
`movespider(5,12,0);`	
`movebug(xb, yb, zb);`	(2, 11, 0)
`movespider(3,11,0);`	
`movebug(xb, yb, zb);`	(1, 10, 0)
`movespider(1,10,0);`	
`Finish;`	

Pentru acest exemplu numărul minim de mutări este 5.

P070012: Pești

Pescarul *Naum* a prins N pești. Greutatea fiecărui pește t_i este măsurată pe țărm. *Naum* lucrează pentru o companie de pește și are dreptul să ia acasă cel mult T grame de pește. El vrea să ia acasă un număr minim de pești, dar care au o greutate totală maximă L(≤T).

Decideți care pești vor fi luați.

Limite

- N≤500000 este un număr întreg pozitiv care reprezintă numărul de pești.
- T≤7000 este un număr întreg pozitv care reprezintă greutatea maximă totală a peștilor care pot fi duși acasă.
- t_i≤35000 este un număr întreg pozitiv care reprezintă greutatea celui de-al i-lea pește.

Timp de execuție: 4 secunde

Fișier de intrare: FISH.IN

Prima linie a fișierului de intrare conține două numere întregi N și T. Fiecare dintre următoarele N linii conține un număr întreg t_i reprezentând greutatea celui de-al i-lea pește.

Fișier de ieșire: FISH.OUT

Dacă nu există o soluție pentru T grame, ieșirea va conține o soluție pentru cel mai mare număr întreg L≤T. prima linie a ieșirii constă într-un număr întreg K reprezentând numărul peștilor care trebuie duși acasă. Fiecare dintre următoarele K linii va conține un număr întreg k_i reprezentând greutatea celui de-al i-lea pește care trebuie dus acasă.

Exemplu

FISH.IN	FISH.OUT
10 280	2
300	200
10	80
30	
80	
200	
20	
20	
60	
10	
100	

CONCURSUL de programare AGORA

Înscrierile pentru participarea la prima rundă s-au încheiat. În continuare vă vom prezenta câteva detalii organizatorice, precum și câteva explicații suplimentare referitoare la regulament.

Modificare e-mail

Dacă nu mai aveți acces la adresa e-mail pe care ați folosit-o pentru a vă înscrie la concurs, atunci va trebui să trimiteți pe adresa concursului *CPA* (**concurs@ginfo.ro** sau **contest@ginfo.ro**) un mesaj care să aibă subiectul `Modificare e-mail` și următoarea structură:

```
ID: codul dumneavoastră de identificare
Nume: numele dumneavoastră
Prenume: prenumele dumneavoastră
Oraș: orașul în care locuiți
Adresă veche: vechea adresă e-mail
Adresă nouă: noua adresă e-mail
```

După trimiterea acestui mesaj, veți primi toate mesajele referitoare la *Concursul de Programare Agora* pe noua adresă indicată.

Obținere ID

Dacă ați uitat care este codul dumneavoastră de identificare puteți să îl aflați trimițând pe adresa concursului *CPA* un mesaj care să aibă subiectul `ID`. Pentru trimiterea acestui mesaj va trebui să folosiți adresa e-mail pe care ați folosit-o pentru înscriere. Dacă nu mai aveți acces la această adresă nu veți putea obține ID-ul și nu veți mai putea participa la concurs.

Informații suplimentare

Pentru informații suplimentare referitoare la *Concursul de Programare Agora* puteți trimite pe adresa concursului un mesaj cu subiectul `Help`.

Abuzuri

Încă din faza înscrierilor am avut "plăcerea" să constatăm că unii nu au altceva mai bun de făcut decât să încerce să se distreze pe seama redacției. Astfel, au fost concurenți care s-au înscris de zeci de ori, "concurenți" care trimiteau "mesaje de înscriere" vide etc. Pentru a evita astfel de evenimente, redacția *GInfo* își rezervă dreptul de a-i descalifica pe acei concurenți care vor trimite, în viitor, mesaje de acest tip. După cum reiese din regulament, dacă trimiteți mai multe mesaje cu soluții, numai primul dintre ele va fi luat în considerare. Totuși, dacă veți trimite un număr mare de astfel de mesaje veți putea fi descalificați.

Descalificări

Mulți concurenți ne-au cerut explicații referitoare la următoarea propoziție din regulament: *Redacția GInfo își rezervă dreptul de a descalifica acei concurenți care trimit soluții identice (sau aproape identice).*

Menționăm faptul că nu vom descalifica nici un concurent dacă nu suntem convinși că acesta a preluat soluția altui concurent. Pentru a scăpa de descalificare, nu este suficient să redenumiți variabilele, să inversați ordinea unor porțiuni din cod sau să rescrieți programul folosind un alt limbaj de programare. Din motive evidente, nu putem preciza care sunt metodele folosite pentru a depista concurenții care preiau soluții unii de la alții. Nu avem pretenția că nu putem greși. Totuși nu putem permite concurenților să nu respecte regulile acestui concurs.

Știm că uneori soluția este foarte simplă și mulți concurenți vor ajunge la ea prin aceeași metodă, iar sursele lor vor fi foarte asemănătoare; în aceste cazuri ei nu vor fi descalificați. Problema se va pune în momentul în care sursele asemănătoare nu vor rezolva corect problema. Este mult mai puțin probabil ca doi concurenți să comită aceeași eroare în rezolvarea unei probleme. În aceste cazuri redacția va avea în vedere descalificare acestora.

Dialog

Nu am avut niciodată pretenția că *GInfo* este o revistă perfectă sau că CPA este un concurs organizat ireproșabil. Totuși cuvintele folosite de unii cititori pentru a semnala unele erori (în majoritatea cazurilor imaginare) au depășit limita bunului simț. Noi am fost întotdeauna deschiși dialogului cu cititorii noștri, cu condiția ca acesta să fie unul civilizat. Dacă cei vizați nu se vor "cuminți" vom recurge la metode mai drastice decât o simplă descalificare.

Runda 2

Vă prezentăm în continuare cele trei probleme de la cea de-a doua rundă a Concursului de Programare Agora. Concurenții care nu s-au înscris până la data de 5 Noiembrie 2000 pot participa la concurs începând cu această rundă dacă trimit mesajul de înscriere până pe data de 5 Decembrie 2000.

CPAR2P1: Acadele*

Nume fișier sursă: `LOL.PAS`, `LOL.C` sau `LOL.CPP`
Nume fișier de intrare: `LOL.IN`
Nume fișier de ieșire: `LOL.OUT`

Descriere: O fabrică de dulciuri produce zilnic N acadele care trebuie ambalate în B cutii, fiecare cutie conținând L acadele. Ciclul de producție durează D zile și satisface următoarele criterii:

- parametrii de producție N, B ($B > 1$) și L sunt aceiași pe toată durata unui ciclu de producție;
- acadelele produse în aceeași zi au ambalaje diferite;
- există exact N tipuri de ambalaje folosite pe toată durata unui ciclu de producție;
- la sfârșitul unui ciclu de producție, pentru fiecare pereche de ambalaje distincte există o singură cutie care conține exact două acadele care au aceste ambalaje, celelalte acadele fiind ambalate diferit față de cele două.

De exemplu, dacă se produc zilnic 9 acadele, ciclul de producție este de 4 zile, atunci în fiecare zi se pot folosi 3 cutii care conțin câte 3 acadele. Cele 9 tipuri de ambalaje pot fi folosite astfel:

	cutia 1	*cutia* 2	*cutia* 3
ziua 1:	123	468	579
ziua 2:	145	278	369
ziua 3:	167	249	358
ziua 4:	189	256	347

Scrieți un program care pentru un număr N dat determină valoarea L, astfel încât valoarea $D - B$ să fie minimă. Dacă există mai multe soluții se va alege cea pentru care valoarea L este maximă.

Date de intrare: Fișierul de intrare conține un singur număr întreg N ($2 \le N \le 2.000.000.000$) reprezentând numărul de acadele produse zilnic.

Date de ieșire: Fișierul de ieșire va conține, pe o singură linie, valorile L, B și D separate printr-un singur spațiu. Dacă nu există soluție se va scrie `0 0 0`.

Exemplu:

`LOL.IN`	`LOL.OUT`
`300000`	`120 2500 2521`

Timp de execuție: 1 secundă/test

* *Problema a fost preluată dintr-o culegere publicată la editura "Computer Libris Agora" din Cluj-Napoca. Din motive ușor de înțeles titlul culegerii va apărea doar după publicarea soluției.*

CPAR2P2: Puncte

Nume fișier sursă: `POINTS.PAS`, `POINTS.C` sau `POINTS.CPP`
Nume fișier de intrare: `POINTS.IN`
Nume fișier de ieșire: `POINTS.OUT`

Descriere: Dându-se n puncte în plan, să se determine o pereche de puncte aflate la distanță minimă unul față de celălalt.

Date de intrare: Fișierul de intrare conține pe prima linie un număr întreg n ($1 \le n \le 10.000$) reprezentând numărul de puncte din plan. Fiecare dintre următoarele n linii conține coordonatele reale x și y ale unui punct cu trei zecimale exacte.

Date de ieșire: Fișierul de ieșire trebuie să conțină pe două linii coordonatele x și y ale celor două puncte cu trei zecimale exacte.

Exemplu:

`POINTS.IN`	`POINTS.OUT`
`3`	`2.000 2.000`
`1.000 1.000`	`1.000 1.000`
`2.000 2.000`	
`10.000 10.000`	

Timp de execuție: 1 secundă/test

CPAR2P3: Determinant

Nume fișier sursă: `DET.PAS`, `DET.C` sau `DET.CPP`
Nume fișier de intrare: `DET.IN`
Nume fișier de ieșire: `DET.OUT`

Descriere: Se dă o matrice pătratică de dimensiune n având elemente numere reale. Să se calculeze determinantul matricei date.

Date de intrare: Fișierul de intrare conține pe prima linie numărul n ($1 \le n \le 200$), iar pe fiecare dintre următoarele n linii câte n numere reale reprezentând o linie a matricei.

Date de ieșire: Fișierul de ieșire trebuie să conțină un număr *întreg* care să indice valoarea determinantului matricei cu o aproximație de două unități.

Exemplu:

`DET.IN`	`DET.OUT`
`2`	`-2`
`1.1 2.2`	Sunt acceptate și soluțiile
`3.3 4.4`	`-4`, `-3` și `-1`

Timp de execuție: 5 secunde/test

GEOMETRIE analitică în C++

Horea Haitonic

Cunoașterea câtorva relații ale geometriei analitice (în plan sau în spațiu) poate ușura mult rezolvarea unor probleme cu caracter practic sau teoretic. Propunem un set de tipuri, constante și funcții care să faciliteze folosirea lor în programare.

Pentru implementarea seturilor de tipuri, constante și funcții folosite pentru rezolvarea unor probleme de geometrie analitică am ales limbajul C++ și programarea orientată obiect, avantajoase prin ușurința în declararea și utilizarea variabilelor. Vom folosi și facilitatea oferită de C++ care permite redefinirea operatorilor după modelul

tip **operator** *simbol_operator(listă_argumente)*

astfel încât lucrul cu primitivele geometrice să fie cât mai intuitiv.

Fișierul antet **GEOM.H** conține declarațiile tipurilor definite (clasele PUNCT, DREAPTA și SEGMENT) precum și prototipurile funcțiilor operator. Fișierul **GEOM.CPP** conține implementarea efectivă a funcțiilor (atât membrii claselor cât și funcțiile operator) definite în fișierul antet. Se adaugă fișierul **GRAF.CPP** conținând implementarea reprezentărilor grafice ale punctelor, dreptelor și segmentelor, grupate separat pentru a permite modificarea lor facilă (ca exemplu, s-a ales varianta cea mai simplă: vom folosi funcțiile BGI pentru a desena pe ecran, suprafața de lucru fiind un pătrat cu vârfurile în (0, 0), (0, 399), (399, 399) și (399, 0); factorul de mărire este precizat printr-o constantă reală pozitivă). Fișierul header va fi inclus în sursă printr-o linie de genul `#include "geom.h"`, iar celelalte două fișiere vor fi grupate într-un proiect împreună cu programul principal (în exemplul din final acesta se găsește în fișierul **TEST.CPP**). Fișierele sursă sunt disponibile pentru download pe site-ul Web al *Gazetei de Informatică* (www.ginfo.ro).

Vom explica pe scurt construcția bibliotecii de tipuri și funcții, rememorând pe parcurs formulele analitice utilizate.

Prima clasă definită este PUNCT, având ca membri de tip dată coordonatele reale ale punctului în planul cartezian iar ca membri de tip funcție doar constructorii:

- `PUNCT(float xx, float yy)` - creează o variabilă de tip PUNCT având în câmpurile `x`, `y` valorile transmise ca argumente. Corpul funcției este extrem de simplu: tot ce avem de făcut este să inițializăm membrii de tip dată cu valorile `xx`, `yy`.
- `PUNCT(void)` - un constructor implicit, fără parametri, este necesar pentru a putea folosi clasa PUNCT în definirea clasei SEGMENT (care conține doi membri de tip PUNCT). Lipsa acestui constructor conduce la apariția erorii `Cannot find default constructor to initialize member...` (a se vedea *Help*-ul C++ pentru mai multe amănunte). Corpul funcției este vid, constructorul implicit fiind necesar doar din punct de vedere sintactic pentru a putea declara variabile de tip PUNCT fără a le inițializa.

O a doua clasă este DREAPTA (definită matematic prin trei numere reale *a*, *b* și *c*, reprezentând coeficienții ecuației generale a dreptei: $ax + by + c = 0$; pentru $a = 0$ obținem o dreaptă orizontală $y = const$, iar pentru $b = 0$ obținem o dreaptă verticală $x = const$). Funcțiile membre sunt:

- Constructorul `DREAPTA(float aa, float bb, float cc)` creează o variabilă de tip DREAPTA având în câmpurile `a`, `b`, `c` valorile transmise prin parametrii `aa`, `bb`, respectiv `cc`. Corpul funcției este analog celui de la constructorul clasei PUNCT.
- Și aici avem un constructor implicit: `DREAPTA(void)`.
- Funcția `float Panta(void)` returnează panta dreptei date prin coeficienții ei *a*, *b*, *c*. Panta este dată de raportul $-a / b$ (dreptele verticale - pentru care $b = 0$ - au panta $\pm\infty$, în funcție de semnul lui *a*).

Următoarea clasă definită este SEGMENT (reprezentat prin două puncte - extremitățile segmentului - care reprezintă datele membre ale acestei clase). Funcțiile membre sunt:

- Constructorul `SEGMENT(PUNCT pct1, PUNCT pct2)` care are rolul de a inițializa datele membre `p1` și `p2` ale clasei cu valorile transmise ca argumente (pentru a putea folosi aici clasa PUNCT avem nevoie de constructorul implicit `PUNCT(void)`, după cum s-a discutat anterior). Corpul funcției constructor al clasei SEGMENT este asemănător cu cel al celorlalți constructori definiți până aici.

- Și pentru această clasă definim un constructor implicit: `SEGMENT(void)`.
- Funcția `Lungime(void)` returnează un număr real reprezentând lungimea segmentului, folosind o formulă bine cunoscută a geometriei analitice: distanța d dintre două puncte $A(x_1, y_1)$ și $B(x_2, y_2)$ este:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Funcția `DREAPTA Suport(void)` returnează dreapta suport a unui segment; în corpul acestei funcții folosim operatorul + (redefinit de noi după cum se va vedea în continuare), care aplicat asupra a două valori de tip `PUNCT` returnează dreapta (unică) care conține cele două puncte date.

Cu aceasta am finalizat prezentarea celor trei clase definite și implementate în biblioteca noastră dedicată geometriei analitice. Urmează să prezentăm operatorii redefiniți pentru variabilele din clasele `PUNCT`, `DREAPTA` și `SEGMENT`, operatori care oferă posibilitatea unei exprimări mai concise și mai clare (comparativ cu folosirea unor funcții "obișnuite" sau a unor macro-uri).

Inițial vom defini egalitatea a două variabile de tip `PUNCT`, `DREAPTA` sau `SEGMENT` prin egalitatea membrilor de tip dată omologi (două puncte sunt considerate egale dacă au abscisele, respectiv ordonatele, egale; două segmente sunt egale dacă punctele care reprezintă extremitățile lor coincid; două drepte sunt egale când coeficienții lor sunt proporționali). Egalitatea a două astfel de primitive geometrice este simbolizată prin operatorul redefinit ==, care poate fi aplicat unor valori ce fac parte din aceeași clasă (`PUNCT`, `DREAPTA`, `SEGMENT`).

Următorul operator redefinit este + care, aplicat asupra a două variabile de tip punct, returnează o variabilă de tip `DREAPTA` care reprezintă dreapta unică ce conține cele două puncte. Acest operator a fost folosit în funcția `SEGMENT::Suport()` (aplicat pentru extremitățile segmentului va returna dreapta suport a segmentului). Pentru a găsi coeficienții a, b, c vom folosi ecuația dreptei care trece prin două puncte $P_1(x_1, y_1)$ și $P_2(x_2, y_2)$:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \Leftrightarrow xy_2 - xy_1 - x_1y_2 + x_1y_1 = x_2y - x_2y_1 - x_1y + x_1y_1$$

sau

$$x(y_2 - y_1) + y(x_1 - x_2) + x_2y_1 - x_1y_2 = 0,$$

de unde rezultă că

$$\begin{cases} a = y_2 - y_1 \\ b = x_1 - x_2 \\ c = x_2y_1 - x_1y_2 \end{cases}.$$

Urmează operatorul `||` care verifică paralelismul a două drepte folosind pantele lor (*două drepte sunt paralele dacă au aceeași pantă*).

Funcția `int operator||(DREAPTA d1, DREAPTA d2)` returnează 1 dacă dreptele sunt paralele și 0 în caz contrar.

Operatorul `!=` verifică perpendicularitatea a două drepte conform regulii binecunoscute: *dreptele sunt perpendiculare dacă produsul pantelor lor este* -1. Sunt considerate perpendiculare și două drepte dintre care una este orizontală (panta este 0) și una este verticală (panta este $\pm\infty$). Corpul funcției este foarte simplu și nu necesită explicații.

Operatorul redefinit `&&` verifică apartenența unui punct la o dreaptă, respectiv la un segment (în funcție de tipul operanzilor, C++ va decide care dintre variante trebuie folosită). Verificarea apartenenței la o dreaptă este simplă: este necesar și suficient ca abscisa și ordonata punctului să verifice ecuația dreptei. Pentru a stabili dacă un punct aparține unui segment, în afară de apartenența la dreapta suport a segmentului (ușor de găsit folosind funcția `SEGMENT::Suport()`) mai trebuie verificată situarea coordonatelor sale între coordonatele extremităților segmentului: dacă P este de coordonate $(x_p; y_p)$ iar extremitățile segmentului sunt $A(x_1, y_1)$ și $B(x_2, y_2)$, este necesar și suficient ca x_p să se afle între x_1 și x_2 și ca y_p să se afle între y_1 și y_2. Cu alte cuvinte trebuie să avem $x_1 \le x_p \le x_2$ sau $x_2 \le x_p \le x_1$ și, respectiv, $y_1 \le y_p \le y_2$ sau $y_2 \le y_p \le y_1$. Pe scurt, aceste condiții pot fi descrise prin inegalitățile $(x_p - x_1)(x_p - x_2) \le 0$ și $(y_p - y_1)(y_p - y_2) \le 0$.

A mai rămas de implementat operatorul * corespunzător intersecției de drepte și segmente (și acesta este implementat în mai multe variante pentru a permite determinarea intersecției dintre două drepte, dintre două segmente sau dintre o dreaptă și un segment). Funcția va returna punctul de intersecție căutat, sau punctul de coordonate (`INFINIT`, `INFINIT`) dacă cele două primitive nu se intersectează.

În cazul intersecției a două drepte, găsirea coordonatelor punctului revine la a rezolva sistemul de ecuații:

$$\begin{cases} a_1x + b_1y = -c_1 \\ a_2x + b_2y = -c_2 \end{cases},$$

unde a_1, b_1, c_1 respectiv a_2, b_2, c_2 sunt coeficienții celor două drepte. Pentru rezolvarea lui vom folosi regula lui *Kramer*. Avem:

$$A = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}$$

și

$$\overline{A} = \begin{pmatrix} a_1 & b_1 & -c_1 \\ a_2 & b_2 & -c_2 \end{pmatrix}.$$

Sistemul este compatibil dacă $d \neq 0$, unde

$$d = \det A = a_1b_2 - a_2b_1.$$

Se observă că propoziția $d \neq 0$ este echivalentă cu $\frac{a_1}{b_1} \neq \frac{a_2}{b_2}$, adică cele două drepte nu sunt paralele.

La început vom verifica existența punctului de intersecție și vom returna punctul de coordonate (`INFINIT`, `INFINIT`) dacă dreptele sunt paralele. După stabilirea compatibilității sistemului, rezolvarea decurge de la sine.

Avem

$$d_x = \begin{vmatrix} -c_1 & b_1 \\ -c_2 & b_2 \end{vmatrix} = b_1c_2 - b_2c_1$$

și

$$d_y = \begin{vmatrix} a_1 & -c_1 \\ a_2 & -c_2 \end{vmatrix} = a_2c_1 - a_1c_2.$$

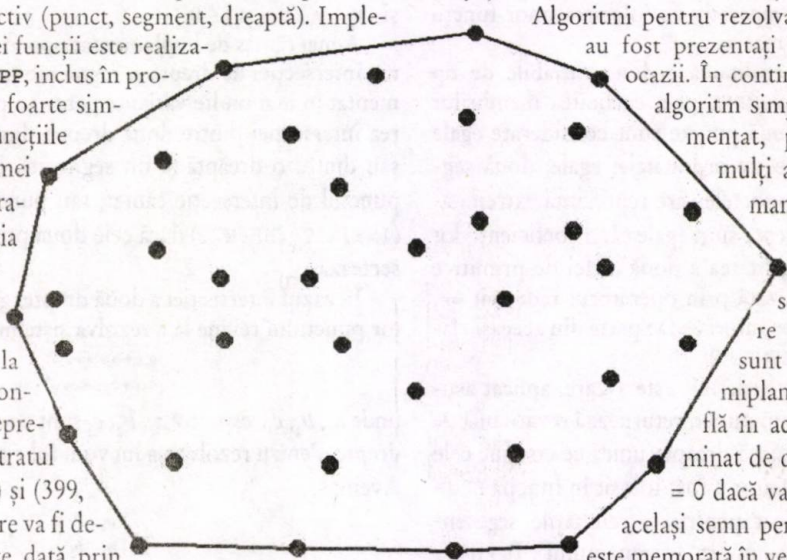
Soluția sistemului este dată de

$$x_p = \frac{d_x}{d}, y_p = \frac{d_y}{d}.$$

Se impune o corecție pentru a evita apariția unor valori de genul `-0.000000`, valori pe care C++ le consideră negative.

Celelalte variante ale operatorului redefinit * se implementează analog: pentru cazul intersecției dintre o dreaptă și un segment se determină întâi punctul de intersecție al dreptei respective cu dreapta suport a segmentului (se folosește în acest sens operatorul definit anterior), iar apoi se face verificarea apartenenței punctului la segment folosind operatorul `&&`. Cazul intersecției a două segmente se tratează asemănător, aici fiind necesară verificarea situării punctului pe ambele segmente. În cazul în care punctul de intersecție nu există este returnat punctul de coordonate (`INFINIT`, `INFINIT`)).

Cele trei clase folosite conțin și câte o funcție membru **`void`** `Af(`**`int`** `c)`, al cărei rol este reprezentarea pe ecran a elementului respectiv (punct, segment, dreaptă). Implementarea acestor trei funcții este realizată în fișierul **`GRAF.CPP`**, inclus în proiect, într-o variantă foarte simplă: se folosesc funcțiile grafice BGI ale firmei *Borland*, în modul grafic `VGAHi` (rezoluția este 640×480, iar adâncimea culorii de 4 biți, deci vom avea $2^4 = 16$ culori la dispoziție). Vom considera ca zonă de reprezentare grafică pătratul determinat de (0, 0) și (399, 399). Culoarea în care va fi desenat elementul este dată prin parametrul `c`. Factorul de scalare `scala` este declarat în fișierul **`GRAF.CPP`** ca variabilă globală externă pentru a putea fi modificat în programul principal în funcție de cerințe, la fel ca și coordonatele ferestrei de reprezentare (variabilele globale `wx1`, `wy1`, `wx2`, `wy2`).



Înfășurătoare convexă

La reprezentarea grafică a primitivelor trebuie făcute câteva precizări:

- Trebuie întotdeauna aplicată transformarea coordonate reale - coordonate ecran (relative la fereastră):

$$x \to (wx1 + wx2)/2 + x \cdot scala;$$

$$y \to (wy1 + wy2)/2 - y \cdot scala.$$

- Se vor folosi funcțiile C++

```
void getviewsettings(struct
                          viewporttype *v)
```

și

```
void setviewport(int x1, int y1,
                 int x2, int y2, int clip)
```

pentru a limita zona de pe ecran în care se poate desena (și anume dreptunghiul (`wx1`, `wy1`, `wx2`, `wy2`)).

- Variabilele globale `wx1`, `wy1`, `wx2`, `wy2` trebuie declarate și în modulul ce conține programul principal folosind clauza **`extern`** pentru a putea fi folosite de funcțiile ce se găsesc în acest fișier.

Încheiem aici prezentarea acestei biblioteci, cu precizarea că ea poate fi extinsă în cel puțin două direcții: adăugarea de noi primitive geometrice (semiplan, cerc, elipsă, poligon etc.) și extinderea într-un spațiu cu trei (sau chiar mai multe dimensiuni).

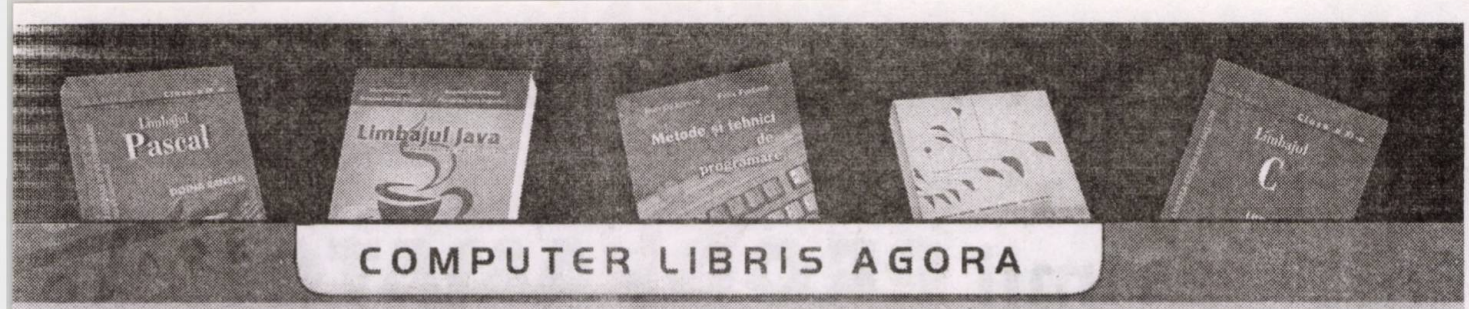
Înfășurătoarea convexă

Pentru exemplificare vom rezolva, folosind funcțiile și tipurile definite, o problemă clasică în care intervin noțiuni de geometrie analitică: *se dau n puncte în planul real și se cere încadrarea lor într-un poligon de perimetru minim*. Aceasta este cunoscută sub numele de *problema determinării înfășurătorii convexe a unei mulțimi de puncte*.

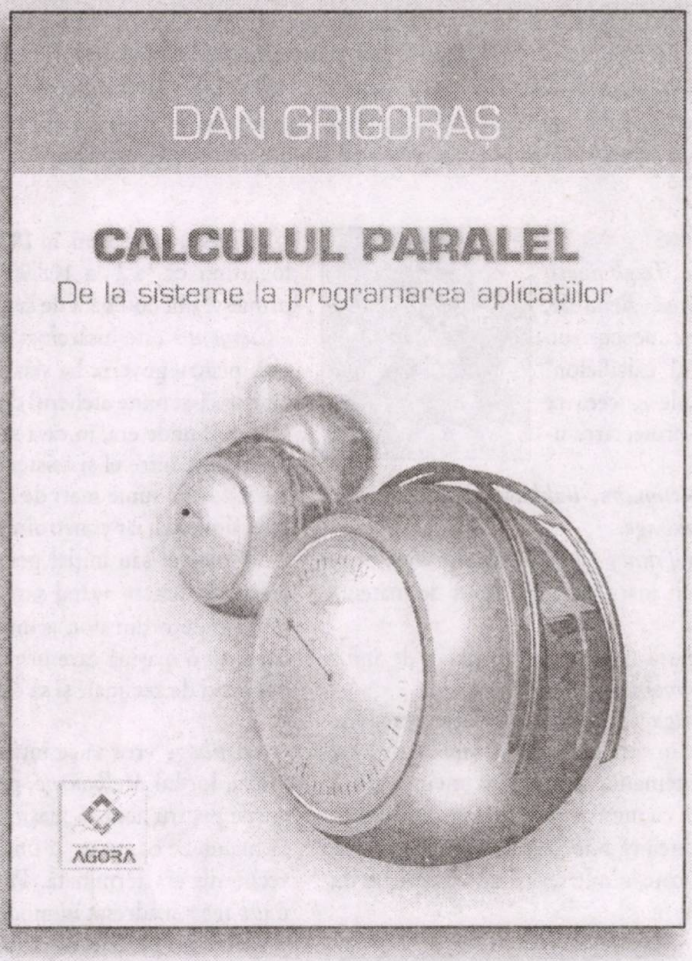
Algoritmi pentru rezolvarea acestei probleme au fost prezentați în ***GInfo*** cu diverse ocazii. În continuare vom descrie un algoritm simplu și ușor de implementat, precizând că există mulți algoritmi mai performanți care rezolvă această problemă. Laturile poligonului vor fi segmentele pentru care toate punctele date sunt situate în același semiplan închis (n puncte se află în același semiplan determinat de dreapta d: $ax + by + c = 0$ dacă valorile $ax_i + by_i + c$ au același semn pentru $i = 1..n$). Soluția este memorată în vectorul `Pol[]` ale cărui elemente sunt de tip `SEGMENT` (am reprezentat un poligon ca o mulțime de segmente ce corespund laturilor). În final se face o afișare a punctelor și a poligonului determinat folosind factorul de scalare `scala`, calculat în funcție de punctele date astfel încât reprezentarea grafică să încapă în fereastra-ecran (`wx1`, `wy1`, `wx2`, `wy2`).

Pentru fiecare segment trebuie să verificăm dacă cele n puncte se află în același semiplan față de dreapta suport a segmentului. Acest lucru poate fi verificat într-un timp $O(n)$. Cele n puncte determină $n(n - 1) / 2$ segmente distincte, deci pentru a determina aceste segmente avem nevoie de un timp $O(n^2)$. Ca urmare, complexitatea algoritmului descris este $O(n^3)$.

Horea Haitonic este elev la Liceul "George Coșbuc" din Năsăud; poate fi contactat prin e-mail la adresa horea_h@hotmail.com.



COMPUTER LIBRIS AGORA



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate, dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performață și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

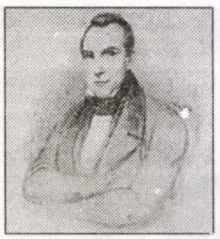
COMPUTER LIBRIS AGORA

Primii PROGRAMATORI

Crina Groșan

Două nume din istoria informaticii... de pe timpuri când nici cuvântul *informatică* nu era cunoscut. Două personalități ale secolului trecut, de numele cărora se leagă noțiunea de *programare*: Charles Babbage și Ada Byron. Se spune că din păcate au "greșit" secolul, deoarece a trebuit să treacă 100 de ani pentru ca ideile și proiectele lor să se poată materializa.

Charles Babbage



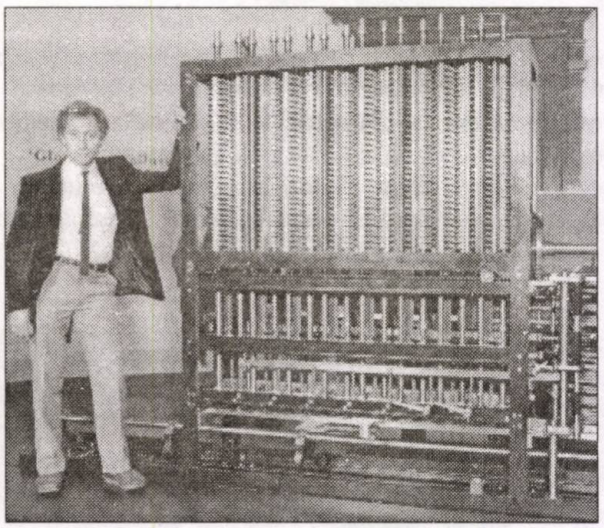
Născut în 1792 în orașul *Teighmouth Devonshire* din *Marea Britanie*, **Charles Babbage** este cunoscut sub numele de "părinte al calculelor" pentru contribuțiile sale în ceea ce privește imaginarea și proiectarea unor mașini de calcul.

Ca și *John von Neumann*, *Babbage* a fost fiul unui bancher, *Benjamin Babbage*.

În 1810 a intrat la *Trinity College*, *Cambridge*, unde, ca și *Newton*, a devenit mai târziu, profesor de matematică.

Babbage este fondatorul *Asociației Britanice* de *Statistică* și a *Societății Astronomice Regale*.

În iulie 1822 *Babbage* îi trimite o scrisoare președintelui de la *Royal Society*, în care îi descrie planul lui de a calcula și tipări tabele matematice cu ajutorul unei mașini. În iunie 1823 se întâlnește cu ministrul de finanțe care îi acordă bani și are convingerea că *Babbage* va face progrese cu această mașină. Din păcate, nimic serios nu s-a realizat după această primă întâlnire.



Totuși, peste 5 ani, în 1827 *Babbage* publică tabelele cu logaritmi de la 1 la 108.000 și ajutat de câțiva prieteni, primește din nou bani de la guvern. Asistentul lui *Babbage* - *Clement* - este însărcinat să construiască mașina proiectată pentru guvern. La sfârșitul anului 1831, *Babbage* hotărăște să-și mute atelierul de construcție a mașinii din hala (grajdul) unde era, în casa sa de pe *Dorset Street*, dar apar divergențe între el și asistent. *Clement* se opune și îi cere lui *Babbage* sume mari de bani. Acesta refuză și *Clement* demisionează, iar construirea mașinii se întrerupe din nou.

Proiectul său inițial pentru mașina de calcul era de a prelucra numere având șase zecimale și de a calcula diferențiale de ordin doi; acum el avea un nou proiect de a construi o mașină care urma să prelucreze numere având douăzeci de zecimale și să calculeze diferențiale de ordinul șase.

Babbage vrea să se întâlnească în 1834 cu primul ministru, lordul *Melbourne*, pentru a-i expune acestuia planurile pentru această mașină nouă. Dar guvernul nu vrea să finanțeze construcția unei mașini noi atât timp cât cea veche nu era terminată. Pe parcursul a opt ani de zile, *Babbage* s-a adresat în mod repetat guvernului, solicitând suspendarea deciziei de a termina mai întâi vechea mașină. Și totuși, *Babbage* cheltuie 17.000 de lire, fără să poată prezenta o mașină de calcul, motiv pentru care încep neplăcerile.

Pe parcursul anului 1842 se duc tratative îndelungate cu ministrul de finanțe și cu primul ministru *Robert Pell*. Chiar dacă aceștia nu doreau să se abandoneze proiectul, în 11 noiembrie 1842, când *Babbage* are întâlnirea finală cu *Pell* i se comunică hotărârea de suspendare a finanțării proiectelor.

În 1851 *Babbage* și-a epuizat toate speranțele de a termina mașina de calcul. Evident, motivul principal părea să fie de ordin financiar, dar unii istorici consideră că eșecul se datorează chiar lui *Babbage*, care era prea nerăbdător și neliniștit în condițiile în care munca pe care dorea să o întreprindă necesita multă răbdare și pasiune. Când vorbea în public, *Babbage*, era de o incoerență aproape notorie.

Nici în fața guvernului nu și-a putut susține cauza, chiar dacă ideile lui erau strălucite. *Pell* l-a lăudat în *Parlament*, dar când a vrut să-i ofere o recompensă pentru munca sa, *Babbage* a refuzat, cerând în schimb titlul de ***pair***.

A murit în 1871 la *Londra*. Chiar dacă mintea sa a avut idei sclipitoare, mult prea avansate pentru epoca în care trăia, *Babbage* a fost puțin cunoscut. Doar *Ducesa de Somerset* a participat la înmormântarea sa. *Societatea Regală* nu a tipărit nici un necrolog; ziarul *Times* l-a ridiculizat.

De fapt, unul din modelele lui *Babbage* a fost realizat și folosit în 1860 pentru întocmirea unor tabele de decese de către o firmă de asigurări. Ceea ce este mult mai important în ceea ce privește realizările lui *Babbage*, se referă la proiectele privind mașina de calcul analitică. În urma lui *Babbage* au rămas mii de desene, schițe, pe baza cărora fiul lui a reușit să construiască efectiv o parte a unității aritmetice. Această mașină ar fi trebuit să aibă o memorie în care ar fi încăput 1000 de numere, fiecare având 50 de cifre zecimale. Pentru introducerea datelor era prevăzută o unitate pentru citirea cartelelor perforate de tipul celor folosite de *Jackard* la controlul războaielor de țesut.

Doar prin prisma istoriei se poate aprecia ce s-a întâmplat de fapt. Mașini de calcul pretențioase, de tipul celei imaginate de *Babbage* nu puteau fi realizate fără electricitate, doar din componente și pe principii mecanice. Trebuia să treacă mult timp - aproximativ 100 de ani - pentru ca tehnica să permită realizarea lor.

Părți din mașina lui de calcul *(Difference Analytical Engine)* au fost acoperite de praf la muzeul *King's College*. Fiul lui degeaba a insistat în scopul continuării cercetărilor, în 1878 *Cayley* a solicitat guvernului să nu se mai ocupe de construirea mașinii lui *Babbage*, urmând ani în care *Babbage* și mașina lui au fost uitate. Abia după anul 1900 *Babbage* a fost repus în drepturi (lângă *Polul Nord* al *Lunii* există un crater care poartă numele lui).

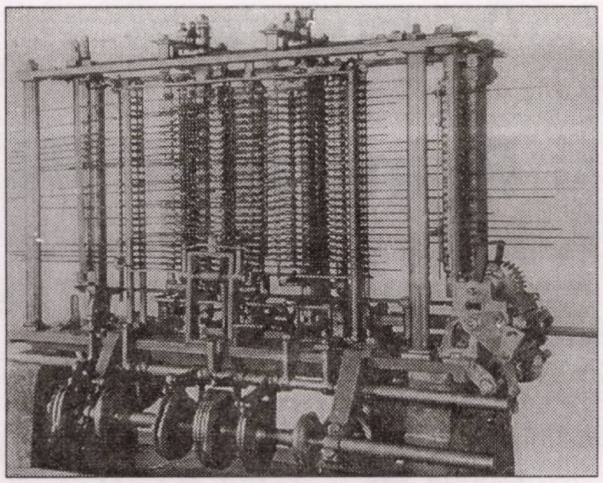
Multe idei ale lui *Babbage* au fost realizate în secolul nostru, de aceea el este socotit ca fiind unul din marii pionieri ai prelucrării datelor.

Ada Byron

Augusta Ada Lovelace (născută ***Byron***) s-a născut la 10 decembrie 1815, în timpul scurtei căsătorii a poetului romantic *Lord Byron* cu *Anne Isabelle Milbanke*. Părinții ei s-au despărțit la doar câteva luni după ce *Ada* s-a născut.

Ada a rămas în grija mamei sale, *Lady Byron*. Aceasta dorea ca fiica ei să nu semene cu poeticul ei tată și a ambiționat educația *Adei* în direcții practice, științifice. *Ada* a avut meditatori la matematică și muzică, discipline opuse "periculoaselor" tendințe poetice.

Complexa inteligență a *Adei* s-a manifestat încă din 1828, când la vârsta de numai 13 ani, și-a surprins prietenii și colaboratorii cu proiectul unei mașini de zbor. Ca studentă a avut înclinații deosebite spre matematică, lucru ieșit din comun în acele vremuri.



Ada l-a întâlnit pe *Charles Babbage* în 1833 tot datorită preocupărilor ei și interesul față de matematică. Au rămas prieteni pentru mult timp și au corespondat îndelungat despre matematică, logică și alte subiecte.

Ada Byron, împreună cu *Charles Babbage* a lucrat la o mașină de calcule matematice (*Difference Engine*) care memora numere și efectua calcule aritmetice simple. *Ada Byron* a explicat proprietățile acestei mașini în articolul ***Observation on Mr. Babbage Analytical Engine*** în 1843. Datorită *Adei* avem cea mai completă descriere a mașinii de calcul a lui *Babbage*.

Proiectele unei mașini mult mai performante constituie primele încercări de a realiza un calculator care să memoreze, prelucreze și să transmită date. *Ada*, dându-și seama că unei asemenea mașini trebuie să i se "spună" ce anume trebuie să facă, a imaginat o serie de programe și astfel ea a intrat în istorie ca inventatoarea programării.

Augusta Ada Byron, contesă de *Lovelace* a murit foarte tânără, în 1852, la vârsta de 37 de ani.

Crina Groșan este absolventă a Facultății de Matematică din cadrul Universității Babeș-Bolyai din Cluj. Poate fi contactată prin intermediul redacției.

Every moment dies a man /
Every moment 1 1/16 is born.
Charles Babbage

A correction to Tennyson's:
"Ev'ry moment a man dies /
Ev'ry moment one is born".

Principiul INDUCȚIEI în GEOMETRIE

Doru Popescu Anastasiu

Metoda inducției matematice are aplicații diverse. Un domeniu în care poate fi folosită este geometria. În continuare vom exemplifica acest lucru prezentând câteva probleme în a căror rezolvare vom aplica principiul amintit.

Aplicația 1

Fie $n \geq 2$ un număr natural și $a_1, a_2, \ldots, a_n$, n numere reale strict pozitive. Să se verifice dacă aceste numere pot fi lungimile laturilor consecutive ale unui poligon. În caz afirmativ se va afișa mesajul DA, altfel mesajul NU.

Exemple

Pentru $n = 3$ și numerele:

```
5 2 1.5
```

se va afișa:

```
NU
```

Pentru $n = 3$ și numerele:

```
2 2 3.5
```

se va afișa:

```
DA
```

Soluție

Se poate demonstra prin inducție matematică (vezi [1] sau [3]) că numerele $a_1, a_2, \ldots, a_n$, pot fi lungimile laturilor unui poligon dacă și numai dacă:

$$a_1 < a_2 + a_3 + \ldots + a_n$$
$$a_2 < a_1 + a_3 + \ldots + a_n$$
$$\ldots$$
$$a_n < a_1 + a_2 + \ldots + a_{n-1}$$

Folosind acest rezultat programul care rezolvă problema este următorul:

Listing APP1.PAS

```
Program Aplicatia_1;
type vector=array[1..100] of Real;
var a:vector;
    i,n:Integer;
    s:Real;
    sw:Boolean;

Begin
  Write('n='); Readln(n);
  for i:=1 to n do
  begin
    Write('a(',i,')=');
    Readln(a[i])
  end;
  s:=0;
  for i:=1 to n do s:=s+a[i];
  sw:=true;
  for i:=1 to n do
    if a[i]>=s-a[i]
      then sw:=false;
  if sw then write('DA')
        else write('NU')
End.
```

Aplicația 2

Se dă un poligon articulat $P_1P_2\ldots P_n$ în plan, pentru care se cunosc lungimile laturilor. Se cere să se deformeze poligonul până se ajunge la un triunghi (eventual degenerat). Se vor preciza:

- vârfurile triunghiului;
- pentru fiecare latură a triunghiului vârfurile poligonului pe care le conține.

Exemplu

Pentru datele de intrare:

```
n=5
P1P2=3
P2P3=10
P3P4=10
P4P5=5
P5P1=12
```

ieșirea va fi:

```
Triunghiul are varfurile: P1P3P5
Latura P1P3 contine punctele: P2
Latura P3P5 contine punctele: P4
```

Soluție

Problema poate fi rezolvată prin inducție matematică.

Fie $P(n)$ propoziția: *"Un poligon cu n laturi poate fi deformat ajungându-se la un triunghi"*, $n \geq 4$.

Pentru un patrulater se aleg două laturi consecutive a căror sumă nu depășește suma celorlalte. Acestea realizează o latură a triunghiului, celelalte două fiind formate din laturile rămase ale patrulaterului. Deci $P(4)$ este adevărată.

Considerăm propoziția adevărată pentru un poligon cu $n+1$ laturi. Alegem două laturi consecutive a căror sumă nu depășește suma celorlalte laturi. Deformăm poligonul astfel încât cele două laturi să ajungă în prelungire. Astfel s-a format un poligon cu n laturi și cum $P(n)$ a fost presupusă adevărată, rezultă că $P(n+1)$ este adevărată.

Deci propoziția $P(n)$ este adevărată pentru orice număr natural $n \geq 4$.

Programul care rezolvă problema, bazându-se pe cele de mai sus, este următorul:

Listing APP2.PAS

```
Program Aplicatia_2;
var l:array[1..50] of Real;
    a:array[1..50,1..50] of Byte;
    d:array[1..50] of Byte;
    i,j,i1,j1,n:Byte;
    s:Real;

procedure interschimbare(var x,y:Byte);
var aux:Byte;
begin
  aux:=x;
  x:=y;
  y:=aux
end;

Begin
  Write('n=');
  Readln(n);
  for i:=1 to n-1 do
  begin
    Write('P',i,'P',i+1,'=');
    Readln(l[i])
  end;
  Write('P',n,'P',1,'=');
  Readln(l[n]);
  l[n+1]:=l[1];
  s:=l[n];
  for i:=1 to n-1 do
  begin
    d[i]:=2;
    a[i,1]:=i;
    a[i,2]:=i+1;
    s:=s+l[i]
  end;
  d[n]:=2;
  a[n,1]:=n;
  a[n,2]:=1;
  l[n+1]:=l[1];
  d[n+1]:=d[1];
  a[n+1,1]:=1;
  a[n+1,2]:=2;
  while n>3 do
  begin
    i:=1;
    while (i<=n) and (s<2*(l[i]+l[i+1])) do
      Inc(i);
      { S-au ales laturile i si i+1 care }
      {          se vor transforma intr-o }
      {                    singura latura }
    if a[i,1]=a[i+1,1] then
      interschimbare(a[i,1],a[i,d[i]]);
    if a[i,1]=a[i+1,d[i+1]] then
    begin
      interschimbare(a[i,1],a[i,d[i]]);
      interschimbare(a[i+1,1],a[i+1,
                                 d[i+1]])
    end;
    for j:=2 to d[i+1] do
      a[i,d[i]+j-1]:=a[i+1,j];
    d[i]:=d[i]+d[i+1]-1;
    l[i]:=l[i]+l[i+1];
    for i1:=i+2 to n do
    begin
      for j1:=1 to d[i1] do
        a[i1-1,j1]:=a[i1,j1];
      l[i1-1]:=l[i1]
    end;
    Dec(n);
    for j:=1 to d[1] do
      a[n+1,j]:=a[1,j];
    l[n+1]:=l[1];
    d[n+1]:=d[1]
  end;
  Writeln('Triunghiul are varfurile:');
  Write('P',a[1,1],'P',a[1,d[1]]);
  if (a[2,1]<>a[1,1]) and
                      (a[2,1]<>a[1,d[1]])
    then Writeln('P',a[2,1])
    else Writeln('P',a[2,d[2]]);
  if d[1]>2 then
  begin
    Writeln('Latura ','P',a[1,1],'P',
          a[1,d[1]],' contine punctele:');
    for j:=2 to d[1]-1 do
      Write('P',a[1,j],' ');
    Writeln
  end;
  if d[2]>2 then
  begin
    Writeln('Latura P',a[2,1],'P',
          a[2,d[2]],' contine punctele:');
```

```
    for j:=2 to d[2]-1 do
      Write('P',a[2,j],' ');
    Writeln
  end;
  if d[3]>2 then
  begin
    Write('Latura P',a[3,1],'P',
          a[3,d[3]],' contine punctele:');
    for j:=2 to d[3]-1 do
      Write('P',a[3,j],' ')
  end
End.
```

Aplicația 3

Se dă un poligon convex $A_1A_2...A_k$ ($k \geq 3$) prin coordonatele vârfurilor sale și $2n$ ($n \geq 1$) puncte distincte prin coordonatele lor situate în interiorul poligonului. Să se împartă poligonul $A_1A_2...A_k$ în $n+1$ poligoane convexe astfel încât toate cele $2n$ puncte să se găsească pe frontierele acestor poligoane.

Exemplu

Pentru $k = 4$, coordonatele vârfurilor poligonului:

```
10 30
20 80
70 60
60 15
20 5
```

$n=2$ și coordonatele punctelor din interiorul poligonului:

```
25 50
40 20
25 20
40 50
```

o soluție posibilă este:

```
(61.11, 20.00) (60.00, 15.00) (20.00, 5.00)
(14.00, 20.00)
(10.00, 30.00) (14.00, 50.00) (67.78, 50.00)
(61.11, 20.00) (14.00, 20.00)
(14.00, 50.00) (20.00, 80.00) (70.00, 60.00)
(67.78, 50.00)
```

Soluție

Problema poate fi rezolvată prin inducție matematică astfel:

Pentru $n = 1$, dreapta determinată de cele două puncte realizează împărțirea cerută.

Presupunem proprietatea adevărată pentru $n - 1$ și fie $2n$ puncte în interiorul poligonului, $n \geq 2$. Fixăm o dreaptă d care nu intersectează poligonul. Fie M punctul cel mai apropiat de d și N punctul ales dintre cele $2n - 1$ puncte, pentru care unghiul format de dreapta MN cu dreapta d, măsurat în sens trigonometric este minim (sau maxim). Dreapta MN împarte poligonul în două poligoane convexe P_1 și P_2, astfel încât P_1 nu conține în interior nici un punct din cele $2n$ puncte, iar P_2 conține în interior cel mult $2n - 2$ puncte. Aplicând ipoteza de inducție deducem că poligonul inițial se împarte în $n + 1$ poligoane convexe, pe ale căror laturi se află toate cele $2n$ puncte.

Programul care rezolvă problema este următorul:

Listing APP3.PAS

```
Program Aplicatia_3;
type punct=record
                x,y:Real
              end;
var a1,a2,a,p:array[1..100] of punct;
    v1,v2:array[1..3] of Integer;
    n,k,i,j,i0,jM,jN,d1,d2,p1,p2,k1,
                                  k2:Integer;
    min,minM,pa,pantaMIN:Real;
    pp1,pp2,pu:punct;
    sw:Boolean;

procedure elimin(i1,i2:Integer);
var j,h:Integer;
    q:array[1..100] of punct;
begin
  h:=0;
  for j:=1 to 2*n do
    if (j<>i1) and (j<>i2) then
    begin
      Inc(h); q[h]:=p[j]
    end;
  for j:=1 to k do
    p[j]:=q[j];
  Dec(n)
end; { elimin }

procedure intersectie(P1,P2,P3,P4:punct;
                                  var P:punct);
var a1,b1,c1,a2,b2,c2:Real;
begin
  a1:=P2.y-P1.y; b1:=P1.x-P2.x;
  c1:=P1.y*(P2.x-P1.x)-P1.x*(P2.y-P1.y);
  a2:=P4.y-P3.y; b2:=P3.x-P4.x;
  c2:=P3.y*(P4.x-P3.x)-P3.x*(P4.y-P3.y);
  P.x:=(b1*c2-b2*c1)/(a1*b2-a2*b1);
  P.y:=(a2*c1-a1*c2)/(a1*b2-a2*b1)
end; { intersectie }

function pozpunct(P1,P2,P:punct):Integer;
var a,b,c:Real;
begin
  a:=P2.y-P1.y;
  b:=P1.x-P2.x;
  c:=P1.y*(P2.x-P1.x)-P1.x*(P2.y-P1.y);
  if a*P.x+b*P.y+c>0 then pozpunct:=1
  else
    if a*P.x+b*P.y+c<0 then pozpunct:=-1
    else pozpunct:=0
end; { pozpunct }
```

```
function panta(P1,P2:punct):Real;
var pa,d:Real;
begin
  d:=Sqrt(Sqr(P1.x-P2.x)+Sqr(P1.y-P2.y));
  panta:=(P2.x-P1.x)/d
end; { panta }

Begin
  Write('k='); Readln(k);
  Writeln('Coord. varfurilor poligonului:');
  for i:=1 to k do
  begin
    Writeln('Punctul ',i);
    Write('abscisa='); Readln(a[i].x);
    Write('ordonata='); Readln(a[i].y)
  end;
  a[k+1]:=a[1];
  Write('n='); Readln(n);
  Writeln('Coord. punctelor din',
                'interiorul poligonului:');
  for i:=1 to 2*n do
  begin
    Writeln('Punctul ',i);
    Write('abscisa='); Readln(p[i].x);
    Write('ordonata='); Readln(p[i].y)
  end;
  min:=a[1].y;
  j:=1;
  for i:=2 to k do
    if min>a[i].y then
    begin
      min:=a[i].y;
      j:=i
    end;
  while n>0 do
  begin
    { Determinare M }
    jM:=1; minM:=p[1].y;
    for i:=2 to 2*n do
      if minM>p[i].y then
      begin
        minM:=p[i].y; jM:=i
      end;
    { Determinare N }
    pantaMIN:=30000;
    for i:=1 to 2*n do
      if i<>jM then
      begin
        pa:=panta(p[jM],p[i]);
        if pa<pantaMIN then
        begin
          pantaMIN:=pa;
          jN:=i
        end
      end;
```

```
    { Gasirea celor doua poligoane }
    { determinate de MN            }
    d1:=0; d2:=0; i0:=0;
    for i:=1 to k do
    begin
      p1:=pozpunct(p[jM],p[jN],a[i]);
      p2:=pozpunct(p[jM],p[jN],a[i+1]);
      if p1*p2<0 then
      begin
        Inc(d1); v1[d1]:=i; i0:=i
      end
      else
        if p1=0 then
        begin
          Inc(d2); v2[d2]:=i
        end;
      if (i0<>0) and (p1<>0) then
      begin
        i0:=i; pu:=a[i]
      end
    end;
    { Dreapta MN intersecteaza poligonul }
    { in interiorul laturilor            }
    if d1=2 then
    begin
      intersectie(p[jM],p[jN],a[v1[1]],
                              a[v1[1]+1],pp1);
      intersectie(p[jM],p[jN],a[v1[2]],
                              a[v1[2]+1],pp2);
      for i:=k downto v1[1]+1 do
        a[i+1]:=a[i];
      a[v1[1]+1]:=pp1; Inc(k);
      for i:=k downto v1[2]+2 do
        a[i+1]:=a[i];
      a[v1[2]+2]:=pp2;
      Inc(k)
    end
    else
      { Dreapta MN intersecteaza o latura }
      { in interior si trece printr-un    }
      { varf al poligonului               }
      if d1=1 then
      begin
        intersectie(p[jM],p[jN],a[v1[1]],
                              a[v1[1]+1],pp1);
        for i:=k downto v1[1]+1 do
          a[i+1]:=a[i];
        a[v1[1]+1]:=pp1;
        Inc(k)
      end;
      k1:=0; k2:=0;
      p2:=pozpunct(p[jM],p[jN],pu);
      for i:=1 to k do
      begin
        p1:=pozpunct(p[jM],p[jN],a[i]);
```

```
        if p1*p2>=0 then
        begin
          Inc(k1); a1[k1]:=a[i]
        end;
        if p1*p2<=0 then
        begin
          Inc(k2); a2[k2]:=a[i]
        end
      end;
      p1:=pozpunct(p[jM],p[jN],pu);
      sw:=true;
      for i:=1 to 2*n do
      begin
        p2:=pozpunct(p[jM],p[jN],p[i]);
        if p1*p2>0 then sw:=false
      end;
      if n=1 then
      begin
        for i:=1 to k1 do
          Write('(',a1[i].x:4:2,',',
                    a1[i].y:4:2,') ');
        Writeln;
        for i:=1 to k2 do
          Write('(',a2[i].x:4:2,',',
                    a2[i].y:4:2,') ');
          Writeln;
        elimin(jM,jN)
      end
      else
        if sw then
        begin
          for i:=1 to k1 do
            Write('(',a1[i].x:4:2,',',
                      a1[i].y:4:2,') ');
          Writeln;
          for i:=1 to k2 do a[i]:=a2[i];
          k:=k2; a[k+1]:=a[1];
          elimin(jM,jN)
        end
        else
        begin
          for i:=1 to k2 do
            Write('(',a2[i].x:4:2,',',
                      a2[i].y:4:2,') ');
          Writeln;
          for i:=1 to k1 do a[i]:=a1[i];
          k:=k1; a[k+1]:=a[1];
          elimin(jM,jN)
        end
  end
End.
```

Aplicația 4

Se dau $8n$ - 4 puncte dispuse sub formă de cruce (vezi figura). În câte moduri putem alege din aceste puncte câte patru, astfel încât ele să fie vârfurile unui pătrat?

Exemplu

Pentru n=3 se va afișa:

21

Soluție

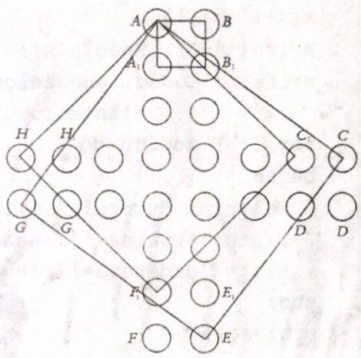
Vom demonstra prin inducție că numărul căutat este $10n$-9.

Pentru n = 1, evident, putem obține 1 = 10 · 1 - 9 pătrate. Se poate dovedi ușor, printr-o simplă numărare, că pentru n = 2 putem alege 10 · 2 - 9 = 11 pătrate.

Considerăm afirmația adevărată pentru $n = k$, unde $k \geq 2$ și o demonstrăm pentru $n = k + 1$.

Considerăm acum figura formată din $8(k + 1)$ - 4 puncte (în desen am considerat k = 3).

Conform ipotezei de inducție putem alege exact $10k$ - 9 pătrate astfel încât fiecare să nu aibă nici un vârf în punctele A, B, C, D, E, F, G, H.



Să considerăm acum pătratele având unul din vârfuri în A. Se observă ușor că există doar trei asemenea pătrate: ABB_1A_1, $ACEG$, AC_1F_1H. În concluzie, există exact zece pătrate cu proprietatea că cel puțin unul dintre vârfuri este unul dintre punctele A, B, C, D, E, F, G, H: patru pătrate congruente cu AC_1F_1H, patru pătrate congruente cu ABB_1A_1 și pătratele $ACEG$ și $BDFH$. Numărul total de pătrate este $(10k - 9) + 10 = 10(k + 1) - 9$.

În continuare propunem cititorului realizarea unui program care să afișeze numărul de pătrate, precum și pătratele, notând vârfurile coliniare cu o anumită literă și folosind indici numerotați de sus în jos și de la stânga la dreapta.

Bibliografie

1. L. Nicolescu etc., *Metode de rezolvare a problemelor de geometrie*, Editura Universității București, 1993
2. L. Panaitopol etc., *Inducția matematică*, Editura GIL, Zalău, 1997;
3. M. Pimsner, S. Popa, *Probleme de geometrie elementară*, Editura Didactică și Pedagogică, 1979
4. Doru Popescu Anastasiu, *Principiul inducției în informatică*, Gazeta de Informatică 3/1998;
5. Doru Popescu Anastasiu, *Principiul inducției în informatică*, Gazeta de informatică 6/1999;
6. W. Sierpinski, *Elementary Theory of Numbers*, Warszawa, 1987.

Dl. prof. Doru Popescu Anastasiu este cadru didactic la Colegiul Național "Radu Greceanu" din Slatina. Poate fi contactat prin e-mail la adresa dopopescu@pcnet.pcnet.ro.

LU-factorizare

Mihai Scorțaru

Prin LU-factorizare se înțelege scrierea unei matrice ca produs între două matrice dintre care una este inferior-triunghiulară (toate elementele de deasupra diagonalei principale au valoarea 0), iar cealaltă este superior-triunghiulară (toate elementele de sub diagonala principală au valoarea 0).

Termenul de *LU*-factorizare poate fi tradus aproximativ prin factorizare jos-sus. Prima matrice are elemente semnificative doar sub și pe diagonala principală, adică în partea "de jos", iar a doua doar deasupra și pe aceasta, adică în partea "de sus". Prefixul *LU* provine din limba engleză, fiind prescurtarea pentru *low up*.

Exemplu

Dându-se matricea

$$A=\begin{bmatrix}1 & 1 & 0\\ 2 & 1 & -1\\ 3 & -1 & 0\end{bmatrix}$$

se poate observa că

$$A=L\cdot U=\begin{bmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & 4 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 1 & 0\\ 0 & -1 & -1\\ 0 & 0 & 4\end{bmatrix}.$$

Ca urmare, matricea A poate fi scrisă ca produs între o matrice inferior-triunghiulară $L=\begin{bmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & 4 & 1\end{bmatrix}$ și una superior-triunghiulară $U=\begin{bmatrix}1 & 1 & 0\\ 0 & -1 & -1\\ 0 & 0 & 4\end{bmatrix}$.

Există și matrice pentru care nu poate fi determinată o *LU*-factorizare. Un astfel de exemplu este matricea

$$A=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}.$$

Determinarea LU-factorizării

O metodă pentru determinarea *LU*-factorizării unei matrice este dată de următorul rezultat matematic:

Teoremă

Fie A o matrice pătratică de dimensiune n și A_k matricele obținute prin tăierea ultimelor n - k linii și coloane ale matricei A. Dacă matricele A_k (k = 1, 2, ..., n - 1) nu sunt singulare (au determinant nenul) atunci există o matrice inferior triunghiulară L care va conține pe diagonala principală doar elemente care au valoarea 1 și o matrice superior-triunghiulară U astfel încât A = LU. Mai mult, această LU-factorizare este unică.

Demonstrație

Vom folosi metoda inducției matematice. Mai întâi vom găsi $L_2=\begin{bmatrix}1 & 0\\ l_{21} & 1\end{bmatrix}$ și $U_2=\begin{bmatrix}u_{11} & u_{12}\\ 0 & u_{22}\end{bmatrix}$ astfel încât $A_2 = L_2U_2$.

Vom avea

$$\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}=\begin{bmatrix}u_{11} & u_{12}\\ l_{21}u_{11} & l_{21}u_{12}+u_{22}\end{bmatrix}$$

deci

$u_{11} = a_{11} \neq 0$

$u_{12} = a_{12}$

$l_{21}=\frac{a_{22}}{a_{11}}$

$u_{22}=a_{22}-\frac{a_{21}}{a_{11}}\cdot a_{12}$

Avem $u_{22}=\frac{\det A_2}{a_{11}}$ deci *LU*-factorizarea este unică.

Vom presupune acum că matricea A_{k-1} are o *LU*-factorizare astfel încât: $A_{k-1} = L_{k-1}U_{k-1}$

$$\begin{bmatrix}A_{k-1} & b_k\\ c_k & a_{kk}\end{bmatrix}=\begin{bmatrix}P_k & 0\\ l_k & 1\end{bmatrix}\cdot\begin{bmatrix}Q_k & u_k\\ 0 & u_{kk}\end{bmatrix}=\begin{bmatrix}P_kQ_k & P_ku_k\\ l_kQ_k & l_ku_k+u_{kk}\end{bmatrix}$$

$$\Rightarrow A_{k-1}=P_kQ_k \underset{\text{unicitate}}{\Rightarrow} \begin{matrix}P_k=L_{k-1}\\ Q_k=U_{k-1}\end{matrix}$$

Obținem $\begin{cases}u_k=L_{k-1}^{-1}\cdot b_k\\ l_k=c_k\cdot U_{k-1}^{-1}\\ u_{kk}=a_{kk}-l_ku_k\end{cases}$.

Pentru $k = n$ existența și unicitatea *LU*-factorizării este dovedită.

Remarcă

Determinantul matricei A este egal cu determinantul matricei U: $\det A = \det U = \prod_{i=1}^{n} u_{ii}$.

Metode de factorizare

Există mai multe metode pentru determinarea LU-factorizării, cele mai cunoscute fiind metoda *Doolittle*, metoda *Crout* și metoda *Choleski*.

Metoda Doolittle

Pentru aplicarea acestei metode trebuie să alegem valoarea 1 pentru elementele de pe diagonala principală a matricei L.

Vom considera matricele

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ l_2 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{n,n-1} & 1 \end{bmatrix};$$

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1,n-1} & u_{1,n} \\ 0 & u_{22} & u_{23} & \dots & u_{2,n-1} & u_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & u_{n-1,n} & u_{n,n} \end{bmatrix}$$

și $A = \begin{bmatrix} a_{11} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n} \end{bmatrix}$ astfel încât $A = LU$.

Această ecuație duce la obținerea unui sistem de ecuații: $\sum_{k=1}^{\min(i,j)} l_{ik} \cdot u_{kj} = a_{ij} \quad i, j = \overline{1,n}$.

Acest sistem poate fi rezolvat aranjând ecuațiile într-o anumită ordine după cum se poate vedea în algoritmul descris în continuare:

```
u11:=a11
pentru j=2,n
   u1j=a1j
   lj1=aj1/u11
sfârșit pentru
pentru i=2,n-1
   uii=aii- Σ(k=1..i-1) lik·uki
   pentru j=i+1,n
      uij=aij- Σ(k=1..i-1) lik·ukj
      lji=(aij- Σ(k=1..i-1) ljk·uki)/uii
   sfârșit pentru
sfârșit pentru
unn=ann- Σ(k=1..i-1) lnk·ukn
```

Din algoritm reiese clar faptul că fiecare element a_{ij} este folosit o singură dată. Aceasta înseamnă că l_{ij} și u_{ij} pot fi memorate în locația pe care o ocupa a_{ij}. Deci, descompunerea este realizată "pe loc"; elementele de pe diagonala principală a matricei L nu mai sunt memorate deloc, deoarece știm că au toate valoarea 1. În concluzie, algoritmul poate fi scris și astfel:

```
pentru j=2,n
   a1j=a1j
   aj1=aj1/a11
sfârșit pentru
pentru i=2,n-1
   aii=aii- Σ(k=1..i-1) aik·aki
   pentru j=i+1,n
      aij=aij- Σ(k=1..i-1) aik·akj
      aji=(aij- Σ(k=1..i-1) ajk·aki)/aii
   sfârșit pentru
sfârșit pentru
ann=ann- Σ(k=1..i-1) ank·akn
```

Metoda Crout

Singura diferență între metoda *Crout* și metoda *Doolittle* este aceea că, în cazul metodei *Crout*, vom alege valoarea 1 pentru elementele de pe diagonala principală a matricei superior-triunghiulare, algoritmul de determinare a LU-factorizării fiind foarte asemănător cu cel folosit pentru metoda *Doolittle*.

Metoda Choleski

Dacă o matrice A este simetrică și pozitiv definită atunci ea are o descompunere specială putând fi scrisă ca un produs dintre o matrice inferior triunghiulară și transpusa acesteia. Matricea inferior triunghiulară nu poate avea pe diagonala principală elemente care să aibă valoarea 0. Uneori această factorizare poartă denumirea de ***extragere a rădăcinii pătrate*** a unei matrice.

Scriind ecuația $A = L \cdot L^t$ se obține:

$$\begin{cases} l_{ii} = \sqrt{a_{ii} - \sum_{i \le k \le n-1} l_{ik}^2} \\ l_{ji} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{i \le k \le n-1} l_{ik} \cdot l_{jk} \right) \end{cases} \quad 1 \le i \le n, i+1 \le j \le n.$$

Algoritmul care realizează această descompunere este descris în continuare:

```
pentru i=1,n
   temp:=aii- Σ(i≤k≤j-1) lik²
```

```
dacă temp>0
  atunci l_ii=temp^1/2
  altfel
    scrie "Matricea nu este pozitiv
        definită! (cu erori de rotunjire)"
    stop
sfârşit dacă
pentru j=i+1,n
```

$$l_{ji} = \frac{1}{l_{ii}}\left(a_{ij} - \sum_{i \le k \le n-1} l_{ik} \cdot l_{jk}\right)$$

```
  sfârşit pentru
sfârşit pentru
```

Aplicație

Cea mai importantă aplicație a *LU*-factorizării este rezolvarea sistemelor liniare de n ecuații cu n necunoscute. Să considerăm sistemul de ecuații $A \cdot X = B$. Dacă scriem $A = L \cdot U$, avem $L \cdot U \cdot X = B$. Vom rezolva sistemul $L \cdot Y = B$, pentru $Y = U \cdot X$, iar apoi vom rezolva sistemul $U \cdot X = Y$.

Exemplu

Pentru sistemul de ecuații

$$\begin{cases} x_1 + x_2 = 4 \\ 2x_1 + x_2 - x_3 = 1 \\ 3x_1 - x_2 = 0 \end{cases}$$

avem

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ 3 & -1 & 0 \end{bmatrix}.$$

Matricea A poate fi *LU*-factorizată astfel:

$$A = L \cdot U = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 4 \end{bmatrix}.$$

$$\Rightarrow L \cdot Y = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 0 \end{bmatrix} = B$$

$$\Rightarrow \begin{cases} y_1 = 4 \\ 2y_1 + y_2 = 1 \\ 3y_1 + 4y_2 + y_3 = 0 \end{cases}$$

$$\Rightarrow \begin{matrix} y_2 = -7 \\ y_3 = 16 \end{matrix}$$

$$\Rightarrow Y = \begin{bmatrix} 4 \\ -7 \\ 16 \end{bmatrix}$$

$$U \cdot X = Y \Rightarrow U \cdot X = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -7 \\ 16 \end{bmatrix} = Y$$

$$\Rightarrow \begin{cases} 4x_3 = 16 \\ -x_2 - x_3 = -7 \\ x_1 + x_2 = 4 \end{cases}$$

$$\Rightarrow \begin{cases} x_3 = 4 \\ x_2 = 3 \\ x_1 = 1 \end{cases}$$

$$\Rightarrow X = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$

Este evident faptul că, dacă am reușit să obținem *LU*-factorizarea matricei A, atunci rezolvarea sistemului de ecuații este foarte simplă.

Complexitate

Determinarea unui element al matricelor L sau U necesită calcularea unei sume care poate avea între 1 și n elemente, deci este nevoie de un timp $O(n)$ pentru determinarea ei. Datorită faptului că avem n^2 elemente, complexitatea algoritmului de determinare a *LU*-factorizării are ordinul de complexitate $O(n^3)$.

Mihai Scorțaru este student în anul III la Universitatea Tehnică din Cluj-Napoca. Poate fi contactat prin e-mail la adresa skortzy@email.ro.

Mențiune

Algoritmul de determinare a LU-factorizării unei matrice, care respectă condițiile din teorema prezentată în cadrul acestui articol, trebuia folosit în rezolvarea unei probleme propuse în cadrul taberei de selecție a echipei care a reprezentat România la Olimpiada Internațională de Informatică din China.

Pentru ca participanții să fie nevoiți să folosească descompunerea "pe loc" a matricei inițiale, dimensiunea maximă a acesteia a fost aleasă în așa fel încât să nu rămână suficientă memorie convențională liberă pentru a putea stoca matrice suplimentare având aceeași dimensiune.

Cel mai mare impediment a fost timpul maxim de execuție admis care nu permitea efectuarea de operații suplimentare față de cele cuprinse în algoritmul prezentat.

Inegalitatea MEDIILOR și calculul RADICALILOR

Lucian Ilea, Iulian Pah

Acest articol își propune să prezinte o metodă de aproximare a radicalilor folosind cunoscuta inegalitate dintre media aritmetică, cea geometrică și cea armonică.

Rădăcina pătrată

Este cunoscută următoarea proprietate:

Dacă $a,b \in \mathbb{R}^*_+$ *și* $m_h = \frac{2}{\frac{1}{a}+\frac{1}{b}}$, $m_g = \sqrt{ab}$, $m_a = \frac{a+b}{2}$, $a \le b$,

atunci $a \le m_h \le m_g \le m_a \le b$ *(inegalitatea mediilor).*

Mai puțin cunoscut, dar ușor de verificat este faptul că $m_g = \sqrt{m_h \cdot m_a}$.

Semnificația acestei inegalități este de restrângere a intervalului inițial $[a, b]$ în care se află $\sqrt{a \cdot b}$. Acest interval se restrânge pas cu pas, conducând, în final, la o aproximare pentru $\sqrt{a \cdot b}$.

Pentru a calcula $\sqrt{x}$, $x \in (0, 1)$ se aplică procedeul de mai sus pentru intervalul inițial $[x, 1]$. Dacă $x > 1$, se calculează $\sqrt{\frac{1}{x}}$ și apoi se inversează.

Această metodă necesită un suport intuitiv simplu, familiarizează elevii cu noțiuni de teorie a erorilor (interesează extragerea radicalului cu o foarte bună aproximație, exactitatea fiind imposibilă), erori inerente în reprezentările în virgulă flotantă. Funcția, descrisă în limbajul *Pascal*, este următoarea:

```
function radical(x:Real):Real;
var a,b,ma,mh:Real;
const epsilon=0.00001;
begin
   a:=x;
   b:=1;
   while b-a>epsilon do
      begin
         ma:=(a+b)*0.5;
         mh:=(2*a*b)/ma;
         b:=ma;
         a:=mh
      end;
   radical:=a
end;
```

Calculele sunt organizate astfel încât numărul de operații de înmulțire și împărțire să fie cât mai mic.

Se poate demonstra și corectitudinea metodei (convergența șirului de intervale). De fapt această metodă nu este decât o deghizare a metodei tangentei, fără a utiliza operația de derivare și cu un criteriu de oprire sugestiv.

Rădăcina de ordinul n

Procedeul poate fi extins pentru calculul lui $\sqrt[n]{x}$ cu $0 < x < 1$. Inegalitatea mediilor se scrie astfel:

$$\frac{n}{\frac{1}{a_1}+\ldots+\frac{1}{a_n}} \le \sqrt[n]{a_1 \cdot \ldots \cdot a_n} \le \frac{a_1+\ldots+a_n}{n}$$

Media geometrică a numerelor $a_1, a_2, \ldots, a_n$ este și media geometrică a numerelor

$$a_1 \cdot \ldots \cdot a_n \cdot \frac{n^{n-1}}{(a_1+\cdots+a_n)^{n-1}}, \frac{a_1+\cdots+a_n}{n}, \ldots, \frac{a_1+\cdots+a_n}{n}$$

Cu excepția primului număr, toate celelalte n - 1 numere sunt media aritmetică a numerelor $a_1, a_2, \ldots, a_n$.

Inițial considerăm $a_1 = x$, $a_2 = 1, \ldots, a_n = 1$. De fapt, din nou, se generează un șir de intervale în care lungimea tinde spre 0.

Avem $\frac{\frac{x+1+\ldots+1}{n}}{x \cdot 1 \cdot \ldots \cdot 1 \cdot \frac{n^{n-1}}{(x+1+\ldots+1)^{n-1}}} = \frac{1}{x}\left(\frac{x+1+\ldots+1}{n}\right)^n < \frac{1}{x}\left(\sqrt{x}\right) = 1$, ceea ce demonstrează convergența spre $\sqrt[n]{x}$.

Bibliografie

Vasile Pop, Tiberiu Oproiu, *Metoda aproximațiilor succesive în matematica elementară*, Lucrările Seminarului de Didactica Matematicii, Universitatea Babeș-Bolyai, vol. 13, 1999

Domnul prof. Lucian Ilea este cadru didactic la Liceul "Emil Racoviță" din Cluj-Napoca, iar domnul lect. univ. Iulian Pah este cadru didactic la Universitatea Babeș-Bolyai din Cluj-Napoca. Pot fi contactați prin intermediul redacției.

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

INTEL vs. AMD

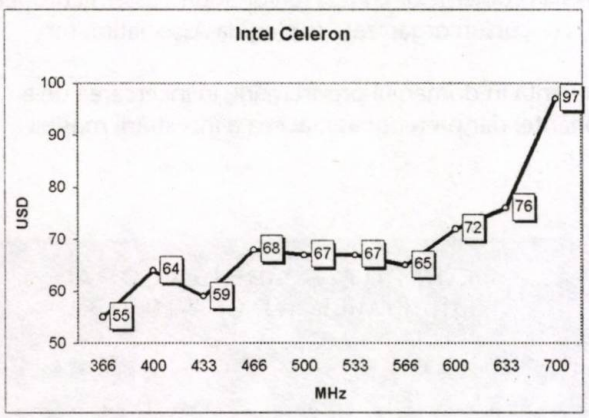
Mihai Scorțaru

În cadrul acestui articol vom studia raportul preț/performanță la principalele procesoare disponibile pe piață. De asemenea, vom prezenta principalele evenimente care au afectat bătălia în ultima perioadă, precum și câteva informații despre ce s-ar putea întâmpla în viitorul apropiat. Toate prețurile menționate în cadrul articolului sunt valabile în Statele Unite ale Americii la începutul lunii noiembrie.

Intel Celeron

Acest procesor destinat calculatoarelor ieftine este momentan disponibil la frecvențe cuprinse între 366 și 700 MHz. Un *Celeron* 366 costă 55$, în timp ce un *Celeron* 700 costă 97$. Chiar dacă frecvența este aproape dublă (cu 91% mai mare), prețul crește doar cu aproximativ 76%. Deocamdată, cel mai convenabil procesor *Celeron* este cel cu frecvența de 566 MHz care costă numai 65$. Acest procesor poate fi foarte ușor forțat să lucreze la 850 MHz, modificând frecvența magistralei de la 66 la 100 MHz. Tot acest procesor a fost protagonistul unui adevărat record în materie de overclocking. Un grup de neozeelandezi au reușit să determine un astfel de procesor să ruleze la 1057 MHz. Acesta a fost primul procesor *Celeron* care a reușit să depășească bariera de 1 GHz. Totuși, costurile materialelor folosite pentru obținerea acestor performanțe au fost foarte ridicate, depășind suma de 1000$. Au fost folosite două galoane de Fluorinert, 30 de litri de azot lichid, cantități enorme de gheață uscată, blocuri mari de aluminiu etc.

Următorul grafic prezintă variația prețurilor procesoarelor *Celeron* în funcție de frecvența la care rulează:



Se observă că, deși ne-am fi așteptat la o creștere liniară a prețului în raport cu frecvența procesorului, graficul arată destul de ciudat, unele procesoare mai rapide fiind mai ieftine decât alte procesoare mai lente.

Principalul motiv pentru care a apărut o astfel de situație este acela că *Intel* nu mai produce procesoare *Celeron* cu frecvențe mici; astfel aceste procesoare sunt greu de găsit și prețul lor crește.

Se preconizează lansarea de procesoare *Celeron* cu frecvențe de 733, 766 și 800 MHz care vor avea magistrala de 66 MHz.

Primul procesor *Celeron* care va avea o magistrală de 100 MHz va apărea la mijlocul anului 2001 și va avea o frecvență de 850 MHz.

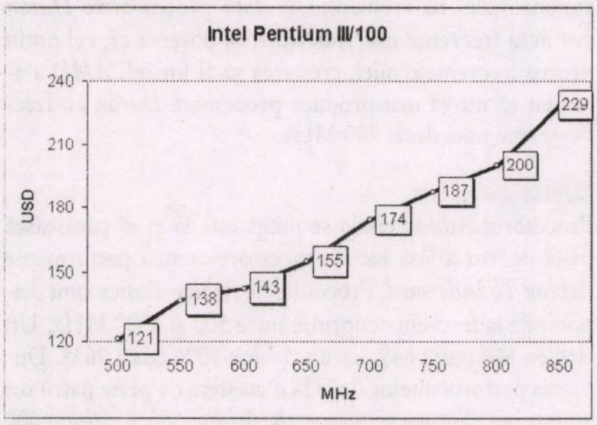
Se spune că acest procesor este de mult timp pe piață sub forma *Celeron* 566, procesor ale cărui performanțe sunt artificial limitate prin folosirea magistralei de 66 MHz. Acest procesor are multiplicatorul 8.5 (66.6... · 8.5 = 566.6...).

Prin modificarea frecvenței magistralei la 100 MHz se obține frecvența 850 = 100 · 8.5, iar modificarea frecvenței magistralei nu reprezintă o problemă pentru cei care au curajul să facă un overclocking.

Intel Pentium III/100

Procesoarele *Pentium III* cu magistrala la 100 MHz se pregătesc să părăsească piața pentru a lăsa locul viitoarelor procesoare *Celeron* care vor folosi o astfel de magistrală. În acest moment puteți achiziționa procesoare de acest tip având frecvențe cuprinse între 500 și 850 MHz. Un *Pentium III* 500/100 costă 121$, iar un *Pentium III* 850/100 costă 229$. La o frecvență cu 70% mai mare, prețul crește cu 89%. Cel mai convenabil procesor din această categorie este *Pentium III* 700/100 care costă 174$. Acesta poate fi forțat să lucreze la 933 MHz prin modificarea frecvenței magistralei de la 100 la 133 MHz.

Graficul următor prezintă variația prețurilor acestor procesoare în raport cu frecvența la care rulează:



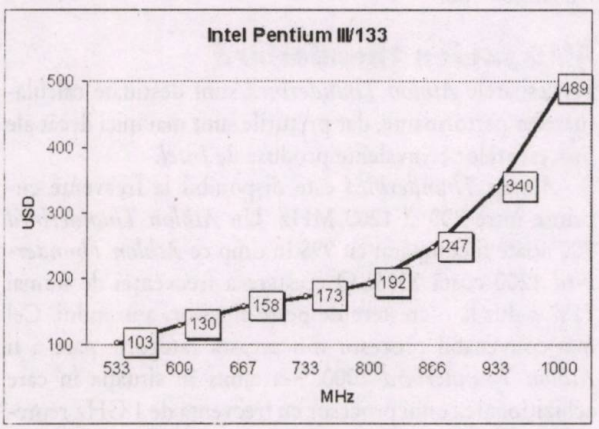
De această dată creșterea este aproape liniară, cea mai mică diferență fiind între 550 și 600 MHz, iar cea mai mare între 800 și 850 MHz. O diferență relativ mare apare și între 500 și 550 MHz.

Intel Pentium III/133

Procesoarele *Pentium III* cu magistrala de 133 MHz sunt, în acest moment, disponibile la frecvențe cuprinse între 533 și 1000 MHz. Procesoarele *Pentium III* sunt destinate calculatoarelor performante, motiv pentru care prețurile lor sunt mai ridicate. Un *Pentium III* 533/133 costă 103$, în timp ce un *Pentium III* 1000/133 costă nu mai puțin decât 489$. La o creștere a frecvenței cu numai 87%, prețul crește de aproape cinci ori. La sfârșitul lunii octombrie *Intel* a redus substanțial prețurile procesoarelor cu frecvențe mari pentru a încerca să contracareze reducerile de prețuri practicate de firma rivală, *AMD*.

Este foarte posibil ca prețurile să scadă și mai mult spre sfârșitul lunii noiembrie când *Intel* va lansa noua familie de procesoare *Pentium 4*. Cel mai convenabil procesor din această categorie pare a fi *Pentium III* 800/133. Forțarea performanțelor acestui tip de procesoare este foarte dificilă datorită frecvenței mari a magistralei.

Dependența prețurilor de frecvență poate fi observată în graficul următor:



Creșterea este aproximativ liniară până la 800 MHz, după care ea devine exponențială. Pentru o creștere de 66 MHz, la frecvențe mari, prețul crește cu 93$ sau 149$.

Din grafic lipsește procesorul *Pentium III* 1133 care a fost retras de pe piață. Relansarea procesorului la această frecvență, folosind o nouă tehnologie, este preconizată pentru al doilea trimestru al anului 2001. În aceeași perioadă va apărea și un procesor *Pentium III* 1266. De asemenea, *Intel* dorește înlocuirea tehnologiei actuale de 0.18 microni cu cea de 0.13 microni.

Intel Pentium 4

Procesoarele *Pentium 4* vor fi lansate oficial, după amânări succesive, la sfârșitul lunii noiembrie a acestui an.

Se pare că de această dată nu vor mai fi întârzieri, deoarece există firme care oferă deja procesorul *Pentium 4* 1.5 GHz. Momentan, prețurile acestor procesoare depășesc suma de 1000$.

Datorită unui contract semnat cu firma *Rambus*, *Intel* este nevoită să producă acest tip de procesoare astfel încât să fie compatibile cu memoria *Rambus*. Aceste memorii sunt foarte scumpe, iar performanțele nu se ridică la nivelul prețurilor.

În prezent *Intel* face eforturi pentru a modifica sau a rezilia contractul cu *Rambus*. Inițial procesoarele *Pentium 4* vor fi lansate la frecvențe de 1.4 și 1.5 GHz. La începutul anului următor procesoarele *Pentium 4* vor avea frecvențe de 1.7 GHz, urmând ca bariera de 2 GHz să fie atinsă la mijlocul anului 2001.

Acest fapt arată că lansarea, în aceeași perioadă, a procesoarelor *Pentium III* 1133 și *Pentium III* 1266 este cel puțin ciudată. Se pare că *Intel* vrea să aibă o portiță de scăpare în cazul în care nu poate evita obligațiile contractuale cu *Rambus*.

La început, procesoarele *Pentium 4* vor fi fabricate pe baza tehnologiei actuale de 0.18 microni, urmând ca, în cursul anului 2001, să se treacă și în cazul acestor procesoare la tehnologia de 0.13 microni.

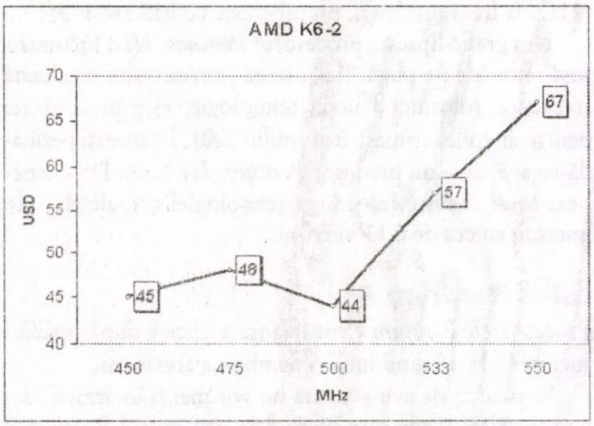
Intel Itanium

Se pare că *Intel* are probleme și cu procesorul pe 64 de biți pe care dorea să îl lanseze în vara anului 2000. Principala dificultate cu care se confruntă compania este stabilirea frecvențelor dorite pentru acest procesor.

AMD K6-2

Procesorul *K6-2*, destinat calculatoarelor ieftine, nu a dispărut încă de pe piață chiar dacă a apărut procesorul *Duron* cu prețuri mai mici și performanțe mai mari. La ora actuală procesoarele *K6-2* sunt disponibile la frecvențe cuprinse între 450 și 550 MHz. Un *K6-2* 450 costă 45$, iar un *K6-2* 550 costă 67$. O creștere a frecvenței cu numai 22% determină o creștere a prețului cu 48%. Nu se poate spune despre nici un procesor K6-2 că este convenabil deoarece un *Duron* 600 costă numai 40$ și este mult mai performant.

Graficul următor prezintă relația dintre frecvență și preț pentru aceste procesoare.



La fel ca și în cazul procesoarelor *Celeron*, graficul are o formă ciudată, datorită faptului că aceste procesoare nu mai sunt în producție.

AMD Duron

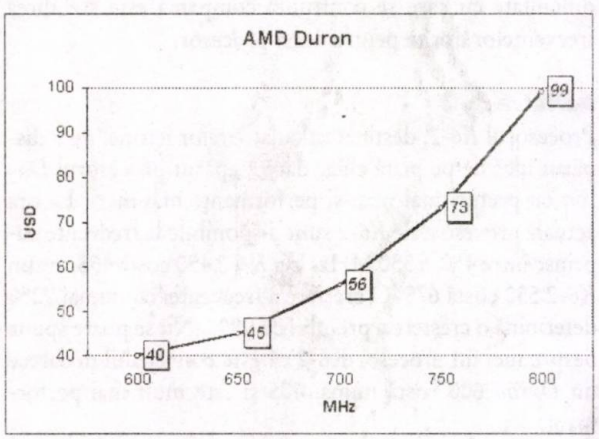
Duron este noul procesor de la *AMD* destinat calculatoarelor ieftine, el fiind înlocuitorul procesorului *K6-2*. *AMD Duron* este disponibil la frecvențe cuprinse între 600 și 800 MHz. Un *Duron* 600 costă 40$ în timp ce un *Duron* 800 costă 99$. O creștere a frecvenței de numai 33% duce la o creștere a prețului cu 147%. Cel mai convenabil procesor din această familie pare a fi *Duron* 700.

Există procesoare *Duron* care nu au multiplicatorul blocat, acestea putând fi forțate să lucreze la frecvențe mult mai mari. De exemplu, schimbarea multiplicatorului de la valoarea 6 la valoarea 9 duce la o creștere a frecvenței de la 600 la 900 MHz.

Din nefericire majoritatea procesoarelor *Duron* au multiplicatorul blocat.

Overclocking-ul prin schimbarea frecvenței magistralei nu duce la creșteri semnificative ale performanțelor. Nu este recomandabil ca frecvența magistralei să fie setată la peste 110 MHz. Datorită arhitecturii folosite această magistrală va fi echivalentă cu una la 220 MHz.

Raportul dintre prețul și frecvența procesoarelor *Duron* poate fi observat în următorul grafic:



Este destul de clar că, deocamdată, creșterea este una exponențială. În momentul în care procesoarele *Duron* vor avea frecvențe mai ridicate vom observa că, cel puțin pentru frecvențele mici, creșterea va fi liniară. *AMD* a anunțat că nu va mai produce procesoare *Duron* cu frecvențe mai mici decât 700 MHz.

AMD Athlon

Procesorul *Athlon* clasic se pregătește și el să părăsească piața pentru a lăsa locul procesoarelor mai performante *Athlon Thunderbird*. Procesoarele *Athlon* clasice sunt disponibile la frecvențe cuprinse între 500 și 1000 MHz. Un *Athlon* 500 costă 64$, iar un *Athlon* 1000 costă 263$. Dublarea performanțelor duce la o creștere de peste patru ori a prețului. Totuși, procesoarele *Athlon* 550 și *Athlon* 600 costă mai puțin decât *Athlon* 500.

Despre nici unul dintre aceste procesoare nu se poate spune că este convenabil. La prețuri comparabile sau chiar mai mici, pot fi achiziționate procesoare *Athlon Thunderbird* echivalente.

Nici overclocking-ul nu duce la performanțe mult mai mari, multiplicatorul fiind blocat, iar modificarea frecvenței magistralei nu duce la creșteri semnificative ale performanțelor.

Graficul următor prezintă raportul dintre frecvență și prețuri pentru acest procesor:

Variația este asemănătoare cu cea de la procesoarele *Celeron* și *K6-2* datorită lipsei de pe piață a anumitor procesoare din această familie. Este interesantă creșterea cu numai 4$ a prețului pentru *Athlon* 1000 față de prețul pentru *Athlon* 950.

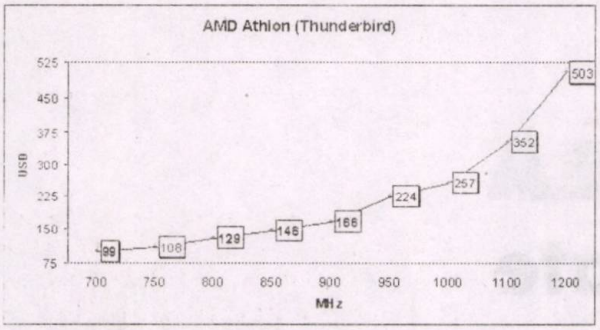
AMD Athlon Thunderbird

Procesoarele *Athlon Thunderbird* sunt destinate calculatoarelor performante, dar prețurile sunt mai mici decât ale procesoarelor echivalente produse de *Intel*.

Athlon Thunderbird este disponibil la frecvențe cuprinse între 700 și 1200 MHz. Un *Athlon Thunderbird* 700 poate fi cumpărat cu 99$ în timp ce *Athlon Thunderbird* 1200 costă 503$. O creștere a frecvenței de numai 71% a dus la o creștere de peste cinci ori a prețului. Cel mai convenabil procesor din această categorie pare a fi *Athlon Thunderbird* 1000. S-a ajuns în situația în care achiziționarea unui procesor cu frecvența de 1 GHz repre-

zintă o investiție rentabilă.

Următorul grafic prezintă variația prețului în raport cu



frecvența pentru aceste procesoare.

Variația este aproape liniară până la frecvența de 1000 MHz, după care creșterea devine exponențială. *AMD* a anunțat faptul că nu va mai produce procesoare *Athlon Thunderbird* cu frecvențe mai mici decât 900 MHz.

AMD a lansat o familie nouă de procesoare *Athlon Thunderbird* care au o magistrală de 266 MHz și folosesc un nou tip de memorie și anume DDR. Potrivit specialiștilor, folosirea acestor memorii a dus la o creștere a performanțelor cu aproximativ 10%. Aceste procesoare au fost lansate pe data de 30 Octombrie 2000 la frecvențe de 1000, 1133 și 1200 MHz. Prețurile oficiale ale acestor procesoare sunt de 385$, 506$, respectiv 673$. Prețurile reale vor fi mai mici cu până la 25%.

Duron vs. Celeron

Piața procesoarelor ieftine este dominată, în acest moment, de procesoarele *Intel Celeron* și *AMD Duron*. Este evident faptul că *AMD* este lider în acest domeniu atât din punct de vedere al performanțelor, cât și din punct de vedere al prețurilor. Totuși numărul de procesoare *Celeron* vândute este mult mai mare decât cel al procesoarelor *Duron*, datorită faptului că majoritatea utilizatorilor preferă să rămână fideli companiei *Intel*.

Athlon vs. Pentium III

Piața procesoarelor performante este dominată de procesoarele *Intel Pentium III* și *AMD Athlon*. Și la acest capitol *AMD* domină din punct de vedere al performanțelor și al prețurilor. Datorită faptului că *Intel* nu produce încă procesoare cu frecvențe de peste 1 GHz, *AMD* rămâne singur pe piață în acest sector. Și numărul procesoarelor *Pentium III* vândute este mai mare decât cel al procesoarelor *Athlon* cu frecvențe sub 1 GHz, motivele fiind aceleași.

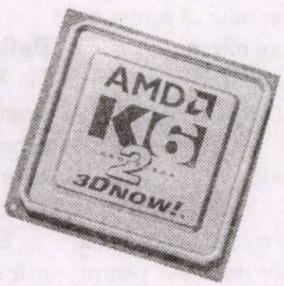
Concluzii

Războiul procesoarelor va continua, cu siguranță, iar singurul câștigător va fi utilizatorul care va avea la dispoziție procesoare din ce în ce mai performante. În același timp, prețurile procesoarelor actuale vor scădea pentru a permite intrarea pe piață a viitoarelor procesoare. *Intel* nu mai poate ignora amenințarea venită din partea *AMD* și nu își mai poate permite erori cum au fost lansarea prematură a procesorului *Pentium III* 1000 și retragerea de pe piață a procesorului *Pentium III* 1133. Lansarea procesorului *Pentium 4* este vitală pentru *Intel* care nu își poate permite ca aceasta să fie un eșec. Există analiști care afirmă că un *Pentium III* la 933 MHz depășește, din punct de vedere al performanțelor, un *Pentium 4* la 1.5 GHz. Dacă acest lucru se dovedește adevărat s-ar putea ca *Intel* să fie nevoită să găsească soluții noi pentru a nu rămâne mult în urma celor de la *AMD*.

Probabil că următorul pas al celor de la *AMD* va fi lansarea procesorului *Athlon* la 1.33 GHz. Acest procesor va apărea spre sfârșitul anului, iar procesoare cu viteze mai mari vor apărea la începutul anului următor. Astfel *Intel* va intra în anul 2001 ca lider din punct de vedere al frecvenței la care lucrează procesoarele, dar, în realitate, se pare că procesoarele *Pentium 4* nu vor fi mai rapide decât noile *Athlon Thunderbird* cu magistrala la 266 MHz. Procesoarele *Intel Pentium 4* au magistrala la 400 MHz; și din acest punct de vedere *Intel* se va afla în fața celor de la *AMD* care nu vor mai modifica prea curând frecvența magistralei.

Se anunță o iarnă interesantă în industria procesoarelor. *Intel* are avantajul renumelui său, iar *AMD* oferă produse calitativ superioare. Nici unul dintre competitori nu își poate permite să facă pași greșiți.

Mihai Scorțaru este student în anul III la Universitatea Tehnică din Cluj. Poate fi contactat prin e-mail la adresa skortzy@email.ro.



PROIECTAREA bazelor de date

Ion Cozac, Mariela Corneanu

Ideea centrală care stă la baza proiectării unei baze de date relaționale este aceea de dependență a datelor. Aceasta se referă la faptul că între atributele unei relații sau între atribute din relații diferite pot exista anumite legături logice (dependențe) și acestea influențează proprietățile relațiilor în raport cu operațiile de actualizare a bazei de date.

Dependențe funcționale

Presupunem că am proiectat o relație cu următoarea structură:

```
CLFACTURI(Nrf,Codc,Numec,Adresa,Dataf).
```

Observăm dependența atributelor `Numec` și `Adresa` față de atributul `Codc`, și de aici rezultă că fiecare valoare a atributului `Codc` determină în mod univoc valoarea corespunzătoare a celorlalte două atribute. Această structură (schemă de relație) introduce o redundanță relativ la atributele `Numec` și `Adresa`, ale căror valori se repetă pentru fiecare factură a aceluiași client. Această redundanță conduce la următoarele anomalii:

- la adăugare: nu se poate înregistra un potențial client decât după ce se emite o factură pentru acesta;
- la ștergere: dacă se șterg toate facturile emise pentru un anumit client se pierd toate informațiile despre acesta; ulterior, dacă acesta cumpără niște produse, informațiile pe care tocmai le-am șters trebuie introduse din nou;
- la modificare: dacă se modifică o informație despre un anumit client (numele, adresa), este necesară parcurgerea întregii relații pentru a actualiza toate aparițiile acestui client; în caz contrar apare pericolul introducerii unei inconsistențe în baza de date datorită faptului că pentru același client sunt înregistrate informații diferite.

Aceste anomalii pot fi evitate dacă se descompune relația `CLFACTURI` în două relații: `CLIENTI` și `FACTURI`, având structurile:

```
CLIENTI(Codc,Numec,Adresa);
FACTURI(Nrf,Codc,Dataf).
```

Un "dezavantaj" al descompunerii efectuate este acela că, pentru a obține informațiile despre un client pentru care am emis o factură, este necesară efectuarea unei operații de cuplare a celor două relații. Operația de cuplare (realizată printr-o instrucțiune `Select` care extrage informații din cele două tabele) nu este atât de costisitoare pe cât ar putea părea la prima vedere, dacă se aleg în mod corespunzător cheile primare și modalitățile de indexare.

După cum rezultă din exemplul de mai sus problema alegerii unui model conceptual corect pentru o bază de date relațională este, de cele mai multe ori, formulată în termenii determinării unor descompuneri pentru scheme de relații date, descompuneri care să izoleze dependențele existente și prin aceasta să se evite anomaliile care decurg din ele.

Definiție

Fie $R(A_1, A_2, ..., A_n)$ o relație, X și Y două atribute (simple sau compuse) submulțimi ale mulțimii de atribute $(A_1, A_2, ..., A_n)$. Atributul X determină atributul Y (sau Y depinde funcțional de X) și notăm $X \rightarrow Y$ dacă și numai dacă orice valoare a atributului X determină în mod unic valoarea atributului Y.

Observații

- dacă $X \rightarrow Y$ atunci pentru orice $Z \subset Y$ avem $X \rightarrow Z$;
- dacă $X \rightarrow Y$ atunci pentru orice $V \subset X$ avem $V \rightarrow Y$.

Definiție

Fie $X \rightarrow Y$ o dependență funcțională; spunem că avem ***dependență totală*** dacă nici o submulțime $V \subset X$ nu induce o dependență funcțională $V \rightarrow Y$; în caz contrar, dacă există o submulțime $V \rightarrow X$ care induce o ***dependență funcțională*** $V \rightarrow Y$, spunem că avem dependență parțială.

Existența în cadrul relațiilor a dependențelor funcționale este un fapt natural. În orice relație există o dependență funcțională a oricărui atribut față de atributul cheie (sau setul de atribute care formează cheia primară): știm că atributul cheie identifică în mod unic fiecare tuplu.

Dependențele funcționale existente în cadrul unei relații se datorează semanticii segmentului din lumea reală care se modelează prin această schemă și reprezintă restricții referitoare la realitatea modelată. Aceste restricții constituie informații asociate relației și care nu pot fi înglobate în reprezentarea relației, deși se reflectă indirect în această reprezentare prin valorile concrete pe care le iau atributele relației. Singura cale de a determina dependențele funcționale din cadrul unei scheme de relație este aceea de a lua în considerare semnificația tuturor atributelor componente.

Primele trei forme normale

Am arătat mai sus că existența anumitor dependențe funcționale în cadrul unei scheme de relație conduce la o serie de anomalii legate de adăugare, ștergere sau modificare. Am văzut că aceste anomalii pot fi evitate dacă se înlocuiesc schemele de relații date prin altele echivalente, în cadrul cărora dependențele sunt supuse anumitor restricții. Aceste scheme de relații se numesc forme normale. Formele normale constituie criterii de ghidare a proiectantului bazei de date în ceea ce privește alegerea schemelor de relație. Regulile de normalizare se aplică cu scopul de a evita anomaliile legate de adăugare, ștergere sau modificare. Efectul direct al normalizării este reducerea redundanței datelor, redundanță care constituie cauza tuturor acestor anomalii. Prin reducerea redundanței se localizează efectul fiecărei operații de actualizare la un număr cât mai restrâns de tuple, ceea ce înseamnă creșterea gradului de independență reciprocă a tuplelor.

În cele ce urmează vom studia doar primele trei forme normale, care se definesc pe baza dependențelor funcționale stabilite între cheia primară și celelalte atribute non-cheie ale unei relații.

În majoritatea problemelor practice aceste trei forme normale sunt mai mult decât suficiente.

Definiție

O relație *R* este în prima formă normală dacă și numai dacă toate atributele sale iau numai valori atomice (scalare).

Această condiție introduce restricția ca domeniile pe care se definesc atributele relației *R* să conțină doar valori atomice, ceea ce înseamnă că toate tuplele unei relații au același număr de câmpuri (aceeași dimensiune).

Toate atributele unei relații trebuie să fie atribute simple, deci nu pot fi atribute compuse, iar valorile din cadrul tuplelor nu pot fi structuri, vectori sau tuple. Modelul relațional nu studiază relații care au tuple de dimensiuni variabile.

A doua și a treia formă normală rezolvă problemele cauzate de existența dependențelor funcționale dintre atributele cheie și cele non-cheie din cadrul tuplelor. Trecerea unei relații din prima formă normală în a doua formă normală în procesul de normalizare a relațiilor constă în izolarea tuturor dependențelor funcționale și crearea unor relații noi definite cu ajutorul atributelor implicate în aceste dependențe. Pentru fiecare dependență funcțională se creează o relație distinctă. Acest lucru este posibil atunci când relația dată are o cheie multiatribut.

Definiție

O relație este în a doua formă normală dacă și numai dacă este în prima formă normală și orice atribut non-cheie este total dependent față de cheia primară a relației.

Pentru orice relație aflată în prima formă normală se poate găsi o descompunere în relații aflate în a doua formă normală care este echivalentă cu relația inițială. Aceasta înseamnă că din relațiile descompunerii se poate reconstitui prin cuplare întreg conținutul de informație al relației inițiale. Așadar înlocuirea unei relații printr-o descompunere a sa în relații aflate în a doua formă normală este un proces reversibil. Avantajul acestei înlocuiri este evitarea unor anomalii legate de operațiile de adaugare, ștergere și modificare. Dar oare sunt evitate toate aceste anomalii?

Exemplu

Fie schema de relație

`PRODFACT(Nrf,Codc,Dataf,Codp,Cant)`

având ca și cheie primară setul `(Nrf,Codp)` și următoarele dependențe funcționale:

- `Nrf→(Codc,Dataf)`;
- `(Nrf,Codp)→Cant`; aceasta deoarece pe o factură putem avea mai multe produse, și un același produs poate apărea pe mai multe facturi.

Deoarece setul `(Codc,Dataf)` depinde total numai de `Nrf`, redundanța are drept consecință o anomalie la adăugare: pentru fiecare poziție de pe factură aceste informații trebuie repetate. Ca dovadă că se poate mai rău decât în exemplul de mai sus, să ne imaginăm ce consecințe ar avea utilizarea unei structuri asemănătoare cu următoarea:

```
TOTFACT(Nrf,Codc,Numec,Adresa,Dataf,
                    Codp,Denp,Pret,Cant).
```

Remarcăm o serie de dependențe funcționale, dintre care prezentăm câteva mai semnificative:

```
Nrf→(Codc,Numec,Adresa,Dataf);
Codc→(Numec,Adresa);
Codp→(Denp,Pret);
(Nrf,Codp)→Cant.
```

Dependența `Nrf→(Numec,Adresa)` se numește tranzitivă, deoarece ea se deduce din dependențele `Nrf→Codc` și `Codc→(Numec,Adresa)`; o ignorăm pentru moment. Să descompunem această schemă astfel:

```
CLFACTURI(Nrf,Codc,Numec,Adresa,Dataf);
PRODUSE(Codp,Denp,Pret);
DETALII(Nrf,Codp,Cant).
```

Se poate verifica faptul că toate cele trei scheme se află în a doua formă normală. Recompunerea relației originale se poate face cu ajutorul unei instrucțiuni `Select SQL`.

Și, totuși, se observă o redundanță supărătoare în schema `CLFACTURI`, datorită atributelor `Numec` și `Adresa`. Aceasta generează aceleași anomalii la adăugare, ștergere sau modificare, deoarece am ignorat dependențele funcționale tranzitive.

Definiție

O relație *R* este în a treia formă normală dacă și numai dacă este în a doua formă normală și toate atributele noncheie sunt total dependente numai de cheia primară și nu și de alte atribute non-cheie.

Să reluăm exemplul de mai sus. Schema `CLFACTURI` se descompune mai departe astfel:

```
CLIENTI(Codc,Numec,Adresa);
FACTURI(Nrf,Codc,Dataf).
```

Indicații

În continuare vom prezenta câteva recomandări referitoare la proiectarea structurii unei baze de date. Analiza problemei pe care trebuie să o rezolvăm identifică toate informațiile care se introduc în baza de date. Se identifică de asemenea toate corelațiile care există între aceste informații - se identifică în fond dependențele funcționale. Pentru a putea identifica mai ușor aceste corelații se recomandă ca în baza de date fiecare entitate care se înregistrează să aibă o cheie de identificare (un cod). Toate informațiile care se deduc dintr-o astfel de cheie (cod) se înregistrează într-o relație asociată clasei respective de entități. Descompunerea unei scheme de relație se face până când toate schemele de relație se află în a treia formă normală.

Exemplu

Analiza unei probleme de evidență a comenzilor a pus în evidență următoarele informații care trebuie înregistrate în baza de date:

- despre furnizori: cod, nume, adresa; fiecare furnizor își prezintă oferta - o listă de produse: cod, denumire, preț;
- un furnizor poate prezenta mai multe produse pe ofertă, un același produs poate apărea pe ofertele mai multor furnizori cu prețuri diferite; toată lumea folosește un sistem unic de codificare a produselor;
- pe o comandă adresată unui furnizor apare: furnizorul, data comenzii, lista produselor comandate (pentru fiecare produs, cantitatea comandată).

Atributele cu care operăm în continuare sunt: `Codf, Numef, Adresa, Codp, Denp, Pret, Nrc, Datac, Cant`.

Dependențele funcționale de bază pe care le-am identificat (am evitat orice dependențe tranzitive) sunt:

`Codf→(Numef,Adresa);`

`Codp→Denp`; prețul nu poate apărea aici deoarece el diferă de la un furnizor la altul;

`(Codf,Codp)→Pret;`

`Nrc→(Codf,Datac);`

`(Nrc,Codp)→Cant.`

Din aceste dependențe funcționale de bază se deduc imediat schemele de relație care vor compune baza de date.

D-na. prof. Mariela Corneanu este cadru didactic la Grupul Școlar de Chimie și Protecția Mediului din Târgu Mureș și poate fi contactată prin e-mail la adresa cmariela@uttgm.ro. *Dl. lect. univ. Ion Cozac este cadru didactic la Universitatea "Petru Maior" din Târgu Mureș și poate fi contactat prin e-mail la adresa* cozac@uttgm.ro.

1.000.000$ pentru Minesweeper

Nu se întâmplă prea des să putem câștiga un milion de dolari analizând un simplu joc. Datorită unei "conjuncturi favorabile" s-ar putea ca acum să avem o mică șansă.

Premiul este oferit de Institutul de Matematică Clay din Cambridge Massachussetes și, pentru a-l câștiga, nu va trebui decât să găsiți un algoritm polinomial pentru a rezolva problema Minesweeper sau să demonstrați că un astfel de algoritm nu există. Probabil că mulți dintre cititori și-au dat deja seama că, de fapt, totul se reduce la a rezolva problema P=NP.

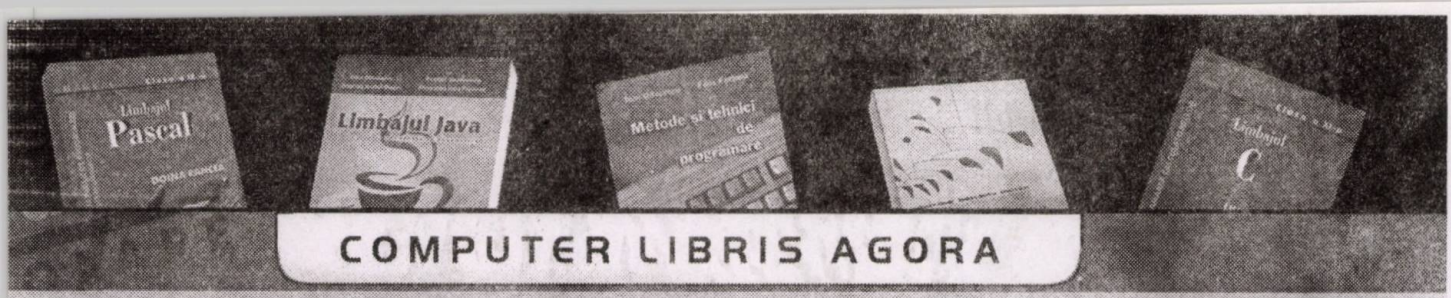
Legătura dintre Minesweeper și această celebră problemă este explicată de Richard Kaye de la Universitatea din Birmingham, Marea Britanie, într-un articol intitulat "Minesweeper is NP-complete" publicat în numărul 22/4 al revistei Mathematical Intelligencer.

Este cunoscut faptul că rezolvarea oricărei probleme NP-complete poate fi redusă la rezolvarea oricărei alte probleme NP-complete. Deci, dacă se descoperă un algoritm polinomial pentru rezolvarea unei probleme NP-complete atunci orice altă problemă NP-completă poate fi rezolvată în timp polinomial.

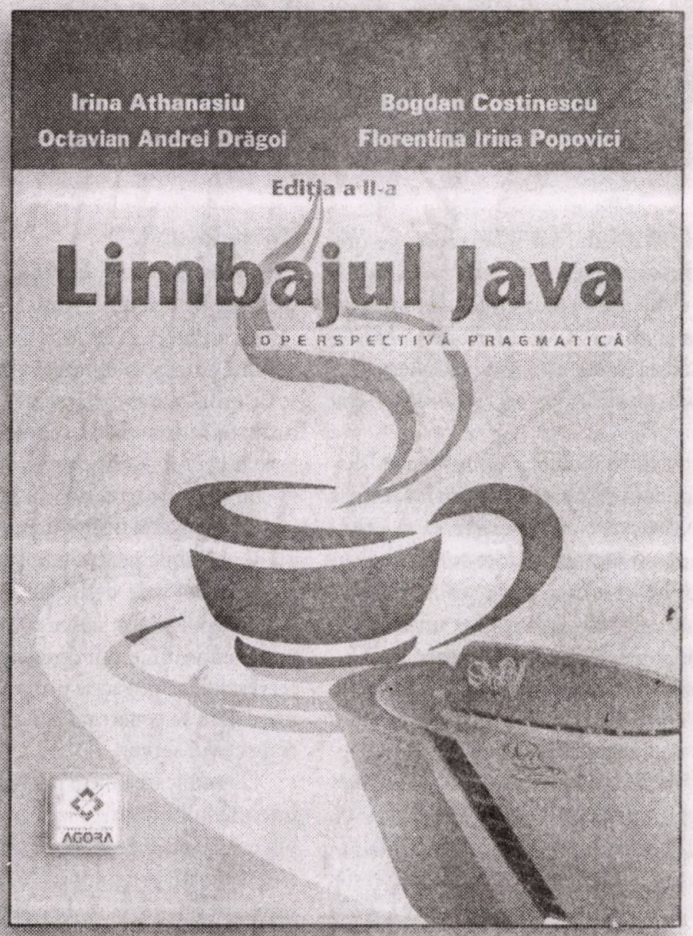
De fapt, problema care trebuie rezolvată este cea a consistenței logice a unei table Minesweeper. Această problemă poate fi enunțată astfel: pentru o configurație dată să se verifice dacă există o posibilitate de amplasare a minelor astfel încât să nu apară contradicții. Kaye a demonstrat că această problemă poate fi redusă la problema satisfiabilității unei formule logice, una dintre primele probleme a cărei NP-completitudine a fost demonstrată. Pentru detalii suplimentare referitoare la teoria NP-completitudinii vă recomandăm să studiați capitolul 36 din cartea "Introducere în algoritmi" scrisă de T.H. Cormen, C.E. Leiserson și R.R. Rivers, a cărei ediție în limba română a apărut la editura Computer Libris Agora din Cluj.

Pentru ca o rezolvare să fie premiată, ea trebuie publicată într-o revistă de referință și trebuie să fie general acceptată de comunitatea matematicienilor în maximum doi ani de la data publicării.

Succes!



COMPUTER LIBRIS AGORA



De ce Java ?

Deoarece tot ce se întâmplă acum în lumea calculatoarelor este pentru Internet, legat de Internet sau accesibil din Internet. Java este limbajul de programare care a devenit un standard de-facto în toate aceste cazuri. Java este acum mai mult decât un limbaj de programare, este o colecție de tehnologii pentru Internet.

De ce reeditată ?

Datorită dinamicii evoluției limbajului, autorii au considerat necesară verificarea exemplelor și pe versiunea Java 2 precum și adăugarea a două noi capitole: componente Java Beans și securitate.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Introducere în CGI

Florin Sabău

Acest articol își propune să fie o inițiere în tainele CGI, o interfață cu ajutorul căreia poate fi realizată interactivitatea între un server și calculatorul pe care sunt vizualizate paginile Web de pe serverul respectiv.

Mulțumită *World Wide Web*-ului (*WWW*), aproape oricine poate pune la dispoziția celorlalți informații într-o formă vizuală plăcută. Probabil că ați navigat de multe ori pe *Web* și v-ați dat seama că acronimele teribile ca *HTTP* sau *HTML* nu sunt altceva decât un mod complicat de a spune *Web* și *modalitatea prin care se prezintă informația pe Web*.

Web-ul s-a dovedit a fi un mediu propice pentru distribuirea informației așa cum se poate observa din imensa popularitate de care se bucură, ca și de creșterea sa exponențială din ultimii ani. Deși unii nu au fost convinși de utilitatea *Web*-ului și au pus creșterea sa pe seama publicității care i se face, acesta este, fără îndoială, o unealtă importantă în accesarea unei palete variate de informații.

Deși pune la dispoziția utilizatorului o interfață foarte prietenoasă (bazată pe conceptul de *hypermedia*), există și alte forme de a distribui eficient date și informații. De exemplu, servicii ca *FTP* (*File Transfer Protocol*) și *Gopher* existau cu mult înaintea apariției *World Wide Web*-ului. De asemenea, e-mail-ul a reprezentat principala modalitate de comunicare pe *Internet* încă de la începuturile acestei rețele. Atunci, cărui fapt se datorează popularitatea *Web*-ului? Nu poate fi ignorat faptul că aspectul multimedia a contribuit mult, dar factorul determinant a fost interactivitatea sa. Pe lângă oferta de informații, fără capabilitatea de a primi informații de la utilizatori, *Web*-ul ar fi un mediu complet static, informația ar fi disponibilă doar în formatul definit de autor. Astfel în loc ca utilizatorul să fie nevoit să răsfoiască mai multe documente pentru a găsi informația dorită (ca și când ar avea în față o carte sau un dicționar), ar fi de preferat ca utilizatorul să testeze câteva cuvinte cheie care să specifice subiectul de interes, iar serverul *Web* să se ocupe de căutarea informațiilor utile.

Deși serverul *Web* poate transmite și primi date, el are o funcționalitate limitată. În mod normal, el nu știe ce să facă cu informațiile pe care utilizatorul le introduce și de cele mai multe ori le ignoră. Pentru a spori capabilitățile serverului *Web*, acesta trebuie "învățat" și aici intră în scenă *CGI*.

Ce este CGI?

Common Gateway Interface (*CGI*) este o interfață între serverul *Web* și utilizator.

La un nivel teoretic, *CGI* sporește funcționalitatea serverului, prin posibilitatea sa de a interpreta datele introduse de utilizatorul care accesează serverul și de a returna informații în funcție de aceste date. Practic, *CGI* este o interfață care-l ajută pe programator să scrie programe care să comunice foarte ușor cu serverul.

CGI pune la dispoziția programatorului o soluție simplă și portabilă pentru a îndeplini aceste deziderate. Există un protocol care definește un mod standard în care programele pot să comunice cu serverul *Web*. Astfel, fără prea multe cunoștințe din domeniul rețelelor sau al arhitecturii serverelor, se pot scrie programe în orice limbaj de programare dacă se generează un cod executabil care să ruleze pe respectivul server.

Comunicarea între program și server se realizează prin intermediul intrării și ieșirii standard (standard input/output). Aceasta înseamnă că tot ceea ce trebuie să știm pentru a scrie un program *CGI* este să citim date de la intrarea standard și să tipărim date la ieșirea standard în limbajul de programare favorit.

Cu excepția interpretării intrării și ieșirii standard, proiectarea și implementarea unei aplicații *CGI* sunt aproape echivalente cu proiectarea și implementarea oricărei alte aplicații. De exemplu, dacă vrem să scriem un program de tip "Hello World!", nu trebuie decât să folosim o funcție de tipul `printf()`, respectând formatul standard definit pentru programele *CGI*.

Ce limbaj să folosesc?

Datorită faptului că *CGI* este o *interfață comună* (*common interface*), nu există limitări asupra limbajului de programare care poate fi folosit. De fapt, poate fi folosit orice limbaj care oferă suport pentru:

- tipărire la ieșirea standard;
- citire de la intrarea standard;
- citirea *variabilelor de mediu* (*environment variables*).

Limbajele se pot împărți în două mari categorii:

- cele care compilează sursele programului într-un fișier obiect; din această categorie fac parte limbajele *C* și *C++* (au ca avantaje un produs final de dimensiuni mici și foarte rapid la execuție, codul compilat este însă dependent de platformă)
- cele care interpretează linie cu linie sursele programului; cel mai folosit limbaj de programare de acest tip este *Perl* (mai lente la execuție, dar sursele sunt portabile; există interpretoare de *Perl* pentru o varietate de platforme și sisteme de operare)

Care dintre cele două limbaje este mai bun? *C* sau *Perl*? Dacă doriți viteză la rulare probabil că veți alege *C*; în schimb, dacă doriți o paletă largă de funcții de procesare a textului sau doriți să scrieți repede un program *CGI*, probabil că *Perl* se va descurca mai bine. De asemenea, un factor important este și experiența anterioară dobândită cu unul dintre ele.

Pentru început vom folosi *C* datorită faptului că este foarte răspândit, existând versiuni pentru majoritatea platformelor și sistemelor de operare.

Având în vedere că nu toți cititorii au acces la un server *Web* și cum cele mai răspândite sisteme sunt cele pe care rulează *Windows*, pentru a încerca exemplele din acest articol recomandăm folosirea aplicației *OmniHTTPd* produsă de firma *Omnicron*. Această aplicație permite simularea comportamentului unui server, folosind un calculator obișnuit.

Primul program CGI

Programarea *CGI* se poate reduce la două sarcini: citirea informațiilor din programul de navigare și tipărirea informațiilor înapoi în programul de navigare.

Probabil că de multe ori, la accesarea unui site *Web*, ați fost nevoit să completați un formular cu datele dumneavoastră personale. În multe cazuri, toate aceste date, după ce se apasă butonul **Submit** (sau un alt buton care are același rol), sunt trimise de programul de navigare către un program *CGI* care rulează pe serverul *Web*. Acesta convertește informațiile într-un format cunoscut de el, le procesează și apoi trimite înapoi programului de navigare (și implicit utilizatorului) rezultatele; acestea pot fi, de exemplu, răspunsurile la o căutare complexă într-o bază de date stocată pe server. Cu alte cuvinte, programarea *CGI* constă în înțelegerea modului în care programul de navigare transmite datele furnizate de utilizator către server și a modului în care programul de navigare se așteaptă să primească rezultatele de la server.

Pentru început vom vedea cum se poate scrie în *C* un program *CGI* foarte simplu (de tip "Hello World!"), urmând ca apoi să explicăm fiecare linie în detaliu.

Va trebuie să introduceți codul sursă și să-l salvați sub numele **HELLO.C**. Acest fișier trebuie compilat, iar fișierul executabil trebuie mutat în directorul cu programe *CGI* de pe server (de obicei `/cgi-bin` pe sistemele *UNIX*). Dacă nu aveți acces la un server *Web* dar ați instalat aplicația *OmniHTTPd*, veți muta executabilul în directorul `c:\httpd\cgi-bin\`. Pentru a vizualiza rezultatul deschideți programul de navigare preferat și introduceți URL-ul http://adresa_server_web/cgi-bin/hello.exe. Dacă folosiți aplicația *OmniHTTPd* sau orice altă aplicație care permite simularea comportamentului unui server *Web* veți introduce *URL*-ul http://localhost/cgi-bin/hello.exe.

Listing HELLO.C

```
#include <stdio.h>
void main(void) {
  printf("Content-Type: text/html\r\n\r\n");
  printf("<HTML>\n")
  printf("  <HEAD>\n");
  printf("    <TITLE>\n");
  printf("      Hello, World!\n");
  printf("    </TITLE>\n");
  printf("  </HEAD>\n");
  printf("  <BODY>\n");
  printf("    <H1>");
  printf("      Hello, World!");
  printf("    </H1>\n");
  printf("  </BODY>");
  printf("</HTML>\n");
}
```

Sunt câteva lucruri care trebuie menționate. În primul rând, am folosit funcții simple de tipărire către `stdout` (*standard output - ieșirea standard*).

În al doilea rând, prima instrucțiune `printf` (cu parametrul `"Content-Type: text/html"`) nu apare la ieșire în programul de navigare deoarece un program *CGI* poate returna orice fel de informație către programul de navigare care trebuie să cunoască tipul informației respective (o pagină *HTML*, o imagine grafică, o secvență audio etc.). Programul de navigare determină acest tip prin intermediul header-ului `"Content Type:"`. În cazul nostru este vorba de o pagină *HTML* (`text/html`). Forma generală a acestui header este:

```
Content-Type: subtype/type
```

unde `"subtype/type"` este un tip *MIME* (*Multipurpose Internet Mail Extensions*) valid. Tabelul 1 prezintă câteva dintre aceste tipuri:

Tabelul 1

Tip MIME	Descriere
`text/html`	*HyperText Markup Language* (*HTML*)
`text/plain`	fișier text simplu
`image/gif`	fișier grafic `.GIF`
`image/jpeg`	fișier grafic `.JPG`
`audio/basic`	fișier sunet `.au`
`audio/x-wav`	fișier sunet `.wav`

Urmează apoi sursa paginii *HTML* propriu-zise. Trebuie remarcat faptul că header-ul și sursa sunt despărțite de o linie liberă (`"\r\n"`).

Pentru a realiza programe *CGI* care să interacționeze cu utilizatorul trebuie să înțelegem mecanismul prin care programul de navigare trimite către programul *CGI* datele introduse de utilizator în formularele *HTML*.

Formularele HTML

Cele mai importante marcaje HTML din punctul de vedere al programatorului de *CGI*-uri sunt <FORM> și <INPUT>. Practic, folosind aceste două marcaje putem realiza orice tip de formular care să permită utilizatorului introducerea datelor în vederea prelucrării de către programul *CGI*. Nu vom realiza o prezentare exhaustivă a celor două marcaje, ci vom sublinia doar elementele mai importante.

Marcajul <FORM> este utilizat pentru a delimita, în cadrul fișierului *HTML*, porțiunea care este folosită de utilizator pentru a introduce date. Atributele marcajului specifică numele programului care va prelucra datele și locația sa (sub forma unui URL), tipul de codificare folosit și metoda cu care se face transferul datelor de intrare către programul *CGI*.

Sintaxa acestui marcaj este:

```
<FORM action="url" method=[post|get]
                  enctype="...">
```

Atributul **enctype** nu este foarte important și de aceea este rar folosit. Este utilizat doar atunci când dorim să trimitem (*upload*) un fișier către server. Atributul **action** conține calea și numele programului *CGI*. După ce utilizatorul apasă butonul `Submit`, toate informațiile din formular sunt codificate și transmise către programul *CGI* prin una din metodele *Post* sau *Get*, selectată prin intermediul atributului **method**.

Marcajul <INPUT> este utilizat pentru a crea câmpuri de editare de text, butoane radio, butoane de selecție etc. Cea mai simplă formă a marcajului <INPUT> este:

```
<INPUT type="tip">
```

unde `tip` poate lua valori ca:

- `text`: un câmp de editare de text (un singur rând); se poate specifica numele acestui câmp (atributul **name**), dimensiunea prin atributul **size**, cât și lungimea maximă (în caractere) prin atributul **maxlength**;
- `submit`: butonul care declanșează operația de codificare a informațiilor și trimitere a lor către programul *CGI*. În mod implicit are valoarea `Submit Query`, dar aceasta poate fi modificată prin intermediul atributului **value**;
- `reset`: butonul a cărui selectare determină resetarea (ștergerea) tuturor câmpurilor;
- `radio`: doar unul din mai multe astfel de butoane poate fi selectat la un moment dat. Atributul **name** este același pentru butoanele din aceeași categorie, variază doar atributul **value**. Pentru ca un astfel de buton să fie selectat implicit, se folosește atributul `checked`;
- `checkbox`: un câmp cu două stări: *selectat* sau *neselectat*; numele și valoarea câmpului este trimisă către programul *CGI* doar dacă respectivul câmp este selectat; pentru a selecta implicit un astfel de câmp se folosește atributul **checked**;
- `hidden`: acest câmp este invizibil și este folosit, de cele mai multe ori, pentru a reține informații de stare;
- `image`: se comportă ca un buton `Submit` doar că încarcă o imagine (URL-ul fișierului care conține imaginea este indicat prin intermediul atributului **src**); prin efectuarea unui click pe imagine se transferă programului *CGI* și coordonatele *x* și *y* ale punctului deasupra căruia se află cursorul mouse-ului.
- `password`: este identic cu `text`, cu excepția faptului că nu se văd caracterele introduse (acestea sunt înlocuite de caractere `'*'` - asterisc).

Metodele GET și POST

În momentul în care utilizatorul a terminat de introdus datele în formular și a apăsat butonul `Submit`, două tipuri de date sunt trimise către programul *CGI*: datele din formular și informații despre programul de navigare și despre server. Cele din cea de-a doua categorie sunt puse la dispoziția programului *CGI* prin intermediul variabilelor de mediu, în timp ce primele sunt disponibile ori printr-o variabilă de mediu (în cazul metodei GET), ori prin intermediul intrării standard (metoda POST).

Indiferent dacă un formular este transmis unui program *CGI* sau nu, orice astfel de aplicație primește informații privitoare la programul de navigare și la server prin intermediul variabilelor de mediu. Aceste informații includ:

- locația programului de navigare în cadrul rețelei;
- tipul programului de navigare și tipul de documente pe care acesta le poate citi;
- numele și versiunea serverului (software) de *Web* care a apelat programul *CGI*;
- instrucțiuni cu privire la modul în care trebuie interpretate datele trimise de programul de navigare.

În continuare vom da câteva exemple de astfel de variabile de mediu:

- variabile generale - `PATH`, `TEMP`, variabile setate din **AUTOEXEC.BAT** sau **CONFIG.SYS**;
- `GATEWAY_INTERFACE` - conține versiunea protocolului *CGI* folosit (de exemplu `"CGI/1.1"`);
- `SERVER_SOFTWARE` - numele software-ului care asigură servicii *HTTP* pe server;
- `SERVER_NAME` - numele serverului *Web*;
- `SERVER_PROTOCOL` - versiunea protocolului *HTTP* (de obicei `"HTTP/1.0"`);
- `REMOTE_HOST` - *hostname*-ul clientului;
- `REMOTE_ADDR` - adresa IP a clientului;
- `REQUEST_METHOD` - poate fi GET sau POST în funcție de metoda folosită;
- `PATH_INFO` - calea către fișierul *CGI* relativă la directorul în care se află rădăcina site-ului *Web*;
- `PATH_TRANSLATED` - calea completă la fișierul *CGI*;
- `QUERY_STRING` - conține datele de intrare din formular transmise prin metoda GET; aceste date sunt codificate;
- `CONTENT_TYPE` - conține un tip MIME care descrie modul în care sunt codificate datele de intrare (implicit are

valoarea "application/x-www-form-urlencoded") și este identic cu atributul **enctype** al marcajului <FORM>;

- CONTENT_LENGTH - conține lungimea datelor de intrare trimise către programul *CGI*. Este definită doar pentru metoda POST;
- HTTP_ACCEPT - conține o listă a tipurilor MIME recunoscute de programul de navigare, separate prin virgulă;
- HTTP_USER_AGENT - conține numele programului de navigare, versiunea și platforma pe care rulează. De obicei are forma: Program_de_navigare/Versiune (Sistem_de_operare);
- HTTP_REFERER - conține URL-ul ultimei pagini vizitate; această variabilă poate fi folosită pentru a insera în pagina generată de programul CGI o legătură spre pagina anterioară.

Pentru a pune în aplicare cele prezentate până acum vom folosi o pagină *HTML* care conține un formular și un program *CGI* care va returna valorile tuturor variabilelor de mediu.

Documentul *HTML* care generează o astfel de pagină *Web* este următorul:

Listing FORMULAR.HTML

```
<HTML>
  <HEAD>
    <TITLE>
      Un formular simplu
    </TITLE>
  </HEAD>
  <BODY>
    <FORM action='http://localhost/
          cgi-bin/formular.exe' method=post>
      <INPUT type=text name='camp'
                        size=15 maxlength=5>
      <BR>
      <INPUT type=radio name=radio
                    value="primul" checked>
      Primul buton <BR>
      <INPUT type=radio name=radio
                      value="al_doilea">
      Al doilea buton <BR>
      <INPUT type=checkbox name=check
                      value="unu" checked>
      Unu <BR>
      <INPUT type=checkbox name=check
                      value="doi" checked>
      Doi <BR>
      <INPUT type=password name=parol>
      <BR>
      <INPUT type=submit name='buton'>
      <INPUT type=reset name='reset'>
    </FORM>
  </BODY>
</HTML>
```

Fișierul sursă pentru script-ul CGI care afișează valorile variabilelor de mediu este următorul:

Listing FORMULAR.C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main(void){
  int l;
  char *s;
  char **p=environ;
  printf("Content-Type: text/html\r\n\r\n");
  while (*p)
    printf("%s <BR>",*p++);
    if (!strcmp(getenv("REQUEST_METHOD"),
                                    "GET"))
      printf("Ati introdus(GET): %s<BR>",
                    getenv("QUERY_STRING"));
    else{
      l=atoi(getenv("CONTENT_LENGTH"));
      s=(char*)malloc(l);
      scanf("%s",s);
      printf("Ati introdus(POST): %s<P>",s);
      printf("<CENTER><H3><A href='%s'> \
                    Inapoi </CENTER> <BR>",
                    getenv("HTTP_REFERER"));
    }
}
```

Fișierul **FORMULAR.HTML** trebuie copiat în rădăcina site-ului *Web* (sau c:\httpd\htdocs dacă folosiți aplicația *OmniHTTPd*), iar fișierul **FORMULAR.C** trebuie compilat, apoi executabilul generat trebuie copiat în directorul cgi-bin (c:\httpd\cgi-bin pentru *OmniHTTPd*).

Din programul de navigare trebuie apelat URL-ul: http://siteweb/formular.html sau http://localhost/formular.html (dacă folosiți un server *Web* local - *OmniHTTPd*).

Codificarea URL

Probabil v-ați dat deja seama că, pentru a trimite date server-ului, programul de navigare le codifică după un anumit format.

Acest format se numește *codificare URL* (*URL encoding*). Aceasta a apărut datorită necesității de a codifica într-un anume fel URL-urile transmise prin e-mail, deoarece prin e-mail pot fi transmise doar primele 128 caractere ale setului de caractere ASCII. O parte din această codificare se referă la caracterele speciale cum ar fi Tab, Space, ghilimelele și, în general, caracterele extinse ale setului ASCII (cu numărul de ordine mai mare decât 127). De ce ar trebui să ne intereseze această codificare? Exsită două motive principale:

- Anumite caractere trebuie codificate dacă sunt incluse într-un URL în cadrul atributului **action** al marcajului <FORM> sau oricare alt câmp trimis către programul

CGI. Spațiile, caracterul procent (%), semnul de întrebare (?) sunt exemple de caractere speciale;
- Toate datele trimise către programul *CGI* sunt codificate și de aceea trebuie înțeles mecanismul de codificare pentru a putea fi decodificate în cadrul scriptului *CGI*.

Referitor la primul aspect, codificarea se face astfel: atunci când programul de navigare întâlnește un caracter special, el îl înlocuiește cu un caracter procent (%) urmat de codul ASCII al caracterului respectiv în hexazecimal (cu două cifre). În tabelul 2 sunt prezentate caracterele speciale care au nevoie de codificare, precum și echivalentul lor hexazecimal.

Tabelul 2

Caracter	Cod ASCII
Tab	09
Space	20
"	22
(	28
)	29
,	2C
.	2E
;	3B
:	3A
<	3C
>	3E
@	40
[	5B
\	5C
]	5D
^	5E
`	60
{	7B
|	7C
}	7D
~	7E

Pe lângă aceste caractere speciale prezentate, alte câteva caractere trebuie codificate deoarece în mod normal au o semnificație specială pentru server:

- Semnul de întrebare (?) se codifică prin %3F; dacă nu este codificat, caracterul indică începerea unui *query*;
- Caracterul ampersand (&) se codifică prin %26; în forma necodificată este separator pentru câmpurile *nume/valoare* într-un *query*;
- Caracterul slash (/) se codifică prin %2F; altfel este folosit pentru a despărți directoarele într-o cale;
- Semnul egal (=) se codifică prin %3D; în forma necodificată desparte valorile *nume* și *valore* într-un *query*
- Diezul (#) se codifică prin %23; este folosit ca referință la o etichetă dintr-un document;
- Semnul procent (%) se codifică prin %25.

Referitor la al doilea tip de codificare folosit pentru datele introduse de utilizator într-un formular spre a fi transmise script-ului CGI, iată câteva reguli care stau la baza acestei codificări:
- Datele sunt transferate ca perechi *nume/valoare*;
- Aceste perechi sunt separate între ele prin caracterul ampersand (&);
- Componenta *nume* este despărțită de *valoare* prin caracterul egal (=). Dacă nu a fost introdusă nici o valoare în formular dar acel câmp are o valoare prestabilită, aceasta va fi folosită. Dacă nu a fost introdusă nici o valoare și nici nu este definită o valoare prestabilită, atunci câmpul *valoare* este vid, dar o pereche *nume/valoare* este totuși transmisă;
- Spațiile conținute în datele introduse în formular sunt caractere speciale și sunt transformate în caractere "+";
- Caracterele speciale nu se pot folosi în URL, ele trebuie codificate;
- Caracterele care au o semnificație specială pentru server (%, ? etc) trebuie codificate;
- Caracterele sunt codificate prin convertirea la scrierea hexazecimală conform tabelului 2.

Florin Sabău este elev în clasa a XII-a la Liceul "Traian Lalescu" din Reșița. Poate fi contactat prin e-mail la adresa fsabau@cs.ro.

Acces gratuit la Internet

Toți abonații Connex pot beneficia de servicii Internet gratuite prin Xnet. Aceștia pot să-și activeze contul Xnet completând informațiile necesare la www.xnet.ro/freexnet.

Chiar dacă va mai trece o perioadă destul de lungă de timp până în momentul în care Internetul va face parte din viața de zi cu zi a românilor, un mic pas a fost făcut. Nu putem decât să ne bucurăm deoarece acum, teoretic, nu mai trebuie să plătim pentru serviciile Internet. La ora actuală există peste un milion de abonați Connex și, din nefericire, în România există doar câteva sute de mii de calculatoare personale. Din această cauză, obținerea unui cont Xnet nu ar trebui să necesite un efort prea mare.

Rămâne în continuare problema tarifelor practicate de RomTeleCom. Dacă și furnizorul român de servicii de telefonie fixă ar urma acest exemplu, poate că Internet-ul ar putea fi apreciat și la noi la adevărata sa valoare. Nu ne rămâne decât să sperăm...

PROTECȚIA programelor

Ștefan Viziru

Securitatea programelor este un domeniu foarte vast cu numeroase teme ce merită discutate. Din multitudinea de probleme pe care le ridică, am ales câteva aspecte mai ușor de înțeles, care nu necesită cunoștințe profunde în domeniu, care ni s-au părut interesante, utile și care țin de cultura informatică.

Firmele producătoare de software din întreaga lume obțin marea majoritate a veniturilor din vânzarea aplicațiilor create. Evident, nu este în interesul lor ca, după vânzarea unei aplicații, cumpărătorul să poată să o multiplice și să o distribuie gratuit. Desigur legea interzice acest lucru, dar punerea în practică a dispozițiilor legale este foarte dificilă. Știm cu toții că în România există foarte puțini utilizatori care folosesc legal produsele soft pe care le "dețin".

Din acest motiv, firmele producătoare de software recurg la protejarea programelor prin parole sau alte mijloace în scopul prevenirii copierii lor neautorizate. Replica nu s-a lăsat așteptată: au apărut *cracker*-ii, persoane care se ocupă cu evitarea mecanismelor de protecție. Nu trebuie confundată noțiunea de *hacker* cu cea de *cracker*.

Probabil că mulți dintre cititori, viitori programatori, vor fi puși în situația de a fi nevoiți să-și protejeze programele create.

La vederea unui fișier executabil mulți au un sentiment de deplină siguranță. Privind simbolurile ASCII cu un editor hex, pare cu neputință ca cineva să poată descifra informațiile conținute în acel fișier. De aceea, când se dorește protejarea unui program mulți cred că e suficient ca acesta să conțină o rutină care cere introducerea unei parole. Dacă parola era corectă, programul putea fi utilizat, dacă nu, execuția programului se încheia.

Acest procedeu oferă o securitate atât de scăzută încât poate să și lipsească, el neconstituind un impediment pentru un *cracker* care deține cunoștințe elementare în acest domeniu.

Din motive evidente, nu putem prezenta modul în care un *cracker* poate trece peste această protecție. Menționăm că scopul nostru nu este acela de a descrie modalități ilegale de a "păcăli" firmele producătoare de software, ci de a vă avertiza că programele trebuie protejate cât mai eficient. Doar cunoscând punctele slabe ale sistemului de securitate, îl veți putea îmbunătăți.

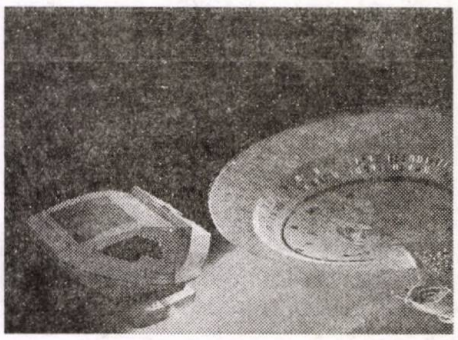
În principiu, o rutină de verificare a parolei va conține o instrucțiune de salt condiționat care va permite continuarea execuției programului dacă parola a fost introdusă corect. Un *cracker* va modifica doar această instrucțiune înlocuind-o cu una de salt necondiționat, astfel încât execuția programului să continue chiar dacă parola nu a fost introdusă corect.

Credem că vă dați cu toții seama că programele comerciale sunt protejate mult mai eficient, prin mecanisme total diferite de această rutină primitivă.

O modalitate de a îmbunătăți puțin protecția, ar fi să vă folosiți de parola introdusă de mai multe ori în program, de exemplu ca parametru de intrare pentru un subprogram. Chiar dacă *cracker*-ul va reuși să treacă peste mecanismul de verificare, programul tot nu va funcționa deoarece subprogramul respectiv nu va funcționa corespunzător.

În concluzie, nu trebuie să ne încredem în metode de parolare care, la prima vedere, par de netrecut. Indiferent ce metodă ingenioasă de protecție am găsi, vor veni alții cu o metodă de evitare a protecției. De aceea, în ceea ce privește securitatea nu strică niciodată să fim puțin neîncrezători și să acționăm conform proverbului: *Cusut dublu ține mai bine!*

Ștefan Viziru este elev în clasa a XII-a la Colegiul "Traian Vianu" din București. Poate fi contactat prin e-mail la adresa hyko_l@usa.net.



STAR TREK
generația Windows

Mulți se întreabă când va sosi momentul în care vom scăpa de sistemul de operare Microsoft Windows. Toată lumea critică acest produs, dar puțini sunt cei care au curajul să nu îl mai folosească deloc. În cele ce urmează vă prezentăm un dialog care ar putea avea loc pe puntea navei stelare U.S.S. Enterprise, dacă acest sistem de operare va supraviețui până în secolul al XXIV-lea.

Lt. Worf: Căpitane, trei nave romulane se decamuflează în fața noastră.

Cpt. Picard: Pe ecran!

Imaginea de pe ecranul principal se modifică, arătând o mulțime de linii orizontale, fiecare cu o lățime de un pixel.

Cpt. Picard: Data, ce se întâmplă aici?

Lt. cmdr. Data: Căpitane, ecranul principal nu are suficientă memorie video pentru a afișa o imagine având această dimensiune. Vă sugerez să alegeți o rezoluție mai mică.

Cpt. Picard: Schimbă rezoluția!

Ecranul clipește și apare o imagine care conține pixeli mari, pătrățoși. Apar în centru trei obiecte care ar putea fi păsări de luptă romulane, dar care, de fapt, seamănă mai mult cu extratereștrii din Space Invaders.

Cpt. Picard: Data, deschide un canal de comunicație cu romulanii.

Lt. cmdr. Data: Da, domnule.

Data ia o clepsidră de pe podea, o întoarce și o așează pe consola din fața sa. Apasă câteva butoane de pe consolă și apoi rămâne nemișcat câteva secunde. Pe ecran apare o rază de lumină care pornește dintr-una dintre navele romulane.

Lt. Worf: Căpitane, se apropie o torpilă fotonică!

Cpt. Picard: Scuturile sus!

Lt. cmdr. Data: Îmi pare rău căpitane, dar încă încerc să execut ultimul ordin pe care mi l-ați dat. Trebuie să vă rog să așteptați până termin, înainte să dați următoarea comandă.

Cpt. Picard: Ce vrei să spui? Data, acest lucru este *important*! Vreau scuturile ridicate *chiar acum*.

Lt. cmdr. Data: Îmi pare rău căpitane, dar încă încerc să execut ultimul ordin pe care mi l-ați dat. Trebuie să vă rog să așteptați până termin, înainte să dați următoarea comandă.

Cpt. Picard: Data, nu ai drivere stocate în memoria ta internă.

Lt. cmdr. Data: Nu am găsit, domnule! Vă rog să introduceți implantul *Setup* #1 în nara mea dreaptă.

Cpt. Picard: Data, nu **am** implantul *Setup* #1.

Lt. cmdr. Data: Nu sunt pregătit pentru a citi nara dreaptă. Anulați, Încercați din nou, Renunțați?

Lt. cmdr LaForge: Permiteți-mi, căpitane. Data, Control-Alt-Delete.

Data ia clepsidra de pe consolă și o pune din nou pe podea.

Lt. cmdr. Data: Romulanii nu răspund chemărilor mele. Apăsați-mi nasul pentru a anula și a vă întoarce în Windows. Trageți de urechea mea stângă pentru a închide acest canal de comunicație care nu răspunde. Veți pierde toate informațiile trimise de romulani.

Lt. cmdr. LaForge merge lângă Data și îl trage de urechea stângă.

Cpt. Picard: Scuturile...

Are loc o explozie violentă. Puntea se cutremură și toți membrii echipajului cad la podea. Din pupitrul lui Wesley Crusher erupe un șuvoi de scântei care îl aruncă pe Wesley departe de pupitru.

Cpt. Picard: ...sus, Data!

Lt. cmdr. Data: Da, domnule!

Lt. cmdr. Riker: Toate punțile, raportul avariilor!

Lt. Worf: Căpitane, slt. Crusher este rănit. Se pare că e inconștient.

Data ia din nou clepsidra, o așează pe consolă și mai apasă câteva butoane. Așteaptă câteva secunde apoi așează din nou clepsidra pe podea.

Lt. cmdr. Data: Scuturile sunt acum ridicate, căpitane.

Cpt. Picard: Cu nici o clipă prea devreme. Worf, țintește cu toate fazerele nava romulană din frunte.

Lt. Worf: Da, domnule!

Cpt. Picard: Domnule Data, preia cârma și pregătește-te pentru manevre evazive.

Lt. cmdr. Data: Îmi pare rău, domnule, dar nu am instalate driverele pentru acea consolă.

Cpt. Picard: Instalează-le!

Lt. cmdr. Data: Vă rog să introduceți implantul *Setup* #1 în nara mea dreaptă.

Cpt. Picard: Numărul Unu, unde păstrăm implanturile *Setup* ale lui Data.

Lt. cmdr. Riker: I le-am dat lui Geordi.

Lt. cmdr. LaForge: Ce!!? Credeam că sunt tot la tine!

Cpt. Picard: Anulează!

Lt. cmdr. Data: Nu sunt pregătit pentru a citi nara dreaptă. Anulați, Încercați din nou, Renunțați?

Cpt. Picard: Atunci renunț!

Lt. cmdr. Data: Nasul curent nu mai este valid!

Data se îndreaptă spre cârmă și apoi apasă câteva butoane. Nava se zguduie, imaginile navelor romulane se mută într-o parte a ecranului și se aude un zgomot asurzitor care pornește dintr-o altă secțiune a navei.

Lt. cmdr. LaForge: Data, ce faci?

Cpt. Picard: Numărul Unu, putem contacta serviciul pentru clienți de la cei care l-au produs pe Data?

Lt. cmdr. Riker: Da, domnule, dar ultima dată când am încercat să-i contactez m-au pus pe linia de așteptare timp de două ore înainte să pot vorbi cu cineva. Și acea persoană nu avea cunoștințe referitoare la androizi ca Data. Era specializată în controlul roboților industriali.

Dintr-o dată se sting toate luminile, ecranul se stinge și zgomotul obișnuit al motoarelor și al ventilatoarelor se oprește. După câteva secunde se aprind luminile roșii de siguranță. Data stă la consolă, absolut nemișcat.

Cpt. Picard: Ce se întâmplă?

Lt. cmdr. LaForge: Lt. cmdr. Data a produs o violare generală de protecție în miezul motoarelor warp.

Licăririle datorate efectului de teleportare apar pe punte și apoi șase romulani în echipament de luptă se materializează pe punte. Un al șaptelea personaj, un ferengi, apare câteva momente mai târziu.

Ferengi: Căpitane, nu vă interesează un sistem Macintosh?

Versiunea în limba engleză a acestui text a fost publicată în revista Computers Magazine, editată de elevii Liceului de Informatică "Tiberiu Popoviciu" din Cluj-Napoca. Traducerea și adaptarea au fost realizate de membrii redacției GInfo.

Previziuni...

HOROSCOP

Berbec — 20 Martie - 19 Aprilie

Luna aceasta veți primi o veste neplăcută: s-ar putea ca hardware-ul calculatorului dumneavoastră să nu vă permită să vă bucurați de ultimele jocuri apărute pe piață. Totuși, nu vă recomandăm să faceți vreun upgrade în această perioadă deoarece s-ar putea să regretați câteva luni mai târziu! Încercați să vă relaxați ascultând muzică; dacă nu aveți încă o colecție impresionantă de MP3-uri, nu vă îngrijorați! Prietenii vă vor ajuta să obțineți ultimele melodii. Bineînțeles, veți încălca legea dreptului de autor, dar asta nu este o problemă pentru dumneavoastră...

Taur — 20 Aprilie - 20 Mai

Bucurați-vă de orice moment de liniște; în ultimul timp ați stat prea mult în fața ecranului care era albastru aproape tot timpul. S-ar putea să ieșiți cu prietenii sau cu familia ceva mai des; este recomandabil ca, în aceste situații, notebook-ul să nu fie prin preajmă. Altfel, ecranele albastre vor apărea tot mai des, numărul operațiilor ilegale pe care le veți efectua va crește și s-ar putea ca mouse-ul și tastatura să nu vă mai asculte. Țineți minte că FreeCell și Solitaire nu sunt metode de relaxare! Dacă tot nu puteți pleca din fața monitorului, încercați Diablo II.

Gemeni — 21 Mai - 20 Iunie

Contrar așteptărilor, Windows-ul nu vă va mai face probleme. Dacă aveați de gând să obțineți venituri suplimentare din tehnoredactare, acum e momentul! Editorul de texte preferat vă va ajuta mult dacă vă veți purta frumos cu el. Încercați să vă controlați impulsul inițial de a apăsa ALT-F4 atunci când ceva nu iese așa cum ați dorit. Totuși, s-ar putea să aveți probleme cu imprimanta. Dacă apar astfel de evenimente neplăcute, nu vă repeziți la cel mai apropiat magazin pentru a cumpăra alta. Mai întâi asigurați-vă că imprimanta are ștecherul în priză.

Rac — 21 Iunie - 22 Iulie

Ar fi momentul să vă dați seama că viața nu înseamnă numai Internet. Site-urile vor fi tot acolo și după ce vă veți întâlni cu persoana iubită. Nu credeți că așteaptă de mult timp un telefon? Probabil că deja s-a săturat de e-mail-urile ocazionale pe care i le trimiteți atunci când vă aduceți aminte. O ieșire în oraș ar putea rezolva problemele cu condiția să nu vă gândiți, în acele momente, la modul în care poate fi instalată ultima versiune de Linux. Dacă, totuși, rămâneți în fața calculatorului, țineți minte că ochii dumneavoastră nu vor rezista o veșnicie...

Leu — 23 Iulie - 22 August

Leii sunt cei mai favorizați în luna aceasta. Ei sunt singurii care știu să îmbine utilul cu plăcutul în sensul că își dau seama exact când este momentul să apese Start/Shut Down. Inteligența și devotamentul îi ajută să-și termine repede treaba, iar spiritul de aventură, care face echipă bună cu simțul umorului, le oferă cele mai plăcute momente din viață petrecute alături de persoana iubită și de familie. Windows-ul poate să aștepte mult și bine pentru că nu îi va mai stresa prea curând chiar dacă se încăpățânează să afișeze ecrane albastre.

Fecioară — 23 August - 22 Septembrie

Având în vedere că nu veți avea timp să vă conectați prea des la Internet, renunțați pentru o vreme la www.ginfo.ro și ocupați-vă cu lucruri mai importante cum ar fi Heroes III sau Quake. Există acolo o mulțime de adversari care așteaptă să fie măcelăriți. Totuși, pentru a vă putea bucura pe deplin de aceste momente ar trebui să vă cumpărați o placă video mai performantă. Și dacă tot ați trecut pe la magazin, schimbați și procesorul. Banii nu vor fi o problemă deoarece vă înțelegeți foarte bine cu membrii familiei.

Balanță — 23 Septembrie - 22 Octombrie

În cazul în care calculatorul refuză să pornească, nu încercați să-l convingeți prin metode proprii. Așteptați puțin și poate se va hotărî să vă facă pe plac. Nu încercați să apelați nici la altcineva deoarece calculatorul este un prieten foarte încăpățânat. Dacă vă enervează prea mult, ar fi bine să îl schimbați. Încercați să vedeți partea bună a lucrurilor! Prietenii vor fi aproape de dumneavoastră și vor încerca să vă ajute să vă rezolvați problemele. Totuși, nu este recomandabil să-i lăsați să încerce să convingă calculatorul să pornească deoarece s-ar putea să fiți nevoiți să sunați la 981.

Scorpion — 23 Octombrie - 21 Noiembrie

Dacă monitorul se stinge pe neașteptate înseamnă că e momentul să vă dați seama că nu aveți ce căuta în fața lui. Nu s-a întâmplat nimic grav; calculatorul a trecut în stand-by deoarece nu ați mai atins nici o tastă în ultima jumătate de oră. Preocupările dumneavoastră nu-și vor găsi rezolvarea dacă rămâneți în fața ecranului. Până la urmă veți ajunge să folosiți mouse-ul pe post de cărămidă sau tastatura pe post de scut. Problemele se vor rezolva dacă vă veți aduce aminte că prietenii nu pot să vă ajute dacă nu știu ce vă frământă.

Săgetător — 22 Noiembrie - 21 Decembrie

Dacă Word-ul crapă a zecea oară înainte să faceți ultima salvare nu înjurați în gând. Înjurați în gura mare că poate aud și cei de la Microsoft. Totuși, gândiți-vă că nu v-a obligat nimeni să învățați să folosiți acest produs! În rest, nimeni nu vă va face probleme. Persoana iubită este alături de dumneavoastră și vă va ajuta să salvați acel fișier. Totuși, ar prefera să închideți calculatorul și se va bucura foarte mult când o veți face. Spre sfârșitul lunii veți petrece câteva zile plăcute la serviciu, poate datorită faptului că se apropie ziua salariului.

Capricorn — 22 Decembrie - 19 Ianuarie

Nu încercați să convingeți calculatorul să facă ceea ce doriți dumneavoastră pentru că nu veți reuși. Țineți minte: a greși este omenesc, dar pentru a zăpăci de tot lucrurile este nevoie de un calculator. Acesta se va pricepe de minune să vă încurce socotelile și singura soluție viabilă de a scăpa de această povară este apăsarea butonului Power. Nu încercați să reporniți calculatorul pentru că rezultatele nu vor fi mai bune decât înainte. Ar fi mult mai bine dacă ați merge la o petrecere la care nu există nici un injformatician printre invitați.

Vărsător — 20 Ianuarie - 17 Februarie

Ați apăsat CTRL+ALT+DEL de prea multe ori în ultima perioadă. Poate ar fi mai bine să lăsați Windows-ul de-o parte pentru un timp. Cu siguranță aveți lucruri mai bune de făcut. S-ar putea ca cineva, care tocmai și-a cumpărat o mașină nouă, să vă invite într-o excursie în afara orașului împreună cu grupul de prieteni. Ar fi bine să acceptați, deoarece nu veți mai rezista prea mult în fața ecranului. Persoana iubită așteaptă de mult timp o astfel de evadare și prietenii nu se vor supăra prea mult dacă veți fi cu unul mai mulți. Încercați totuși să nu forțați nota!

Pești — 18 Februarie - 19 Martie

Nu veți afla niciodată cine este admiratorul/admiratoarea secret/secretă dacă vă încăpățânați să discutați cu necunoscutii de pe chat în speranța găsirii perechii potrivite. Un buchet de flori valorează cât un million de e-mail-uri primite de la persoane aflate la mii de kilometri distanță. O clipă petrecută alături de persoana iubită este mai importantă decât o noapte pierdută în luptă cu adversarii conduși de "prietenii" de pe Internet. Oricum, totul se va rezolva atunci când veți primi factura telefonică.

Acum ai propria ta forță de reacție rapidă!

Acum poți să faci față oricăror situații neprevăzute. Acum ești înarmat cum trebuie. Noul Armada M300 cu procesor mobil Intel® Pentium® III 600 MHz cu tehnologie SpeedStep™ îți dă toată puterea de foc necesară: display de până la 12,1"; hard-disc 12 GB; numai 2,2 cm grosime; modem 56K Mini-PCI integrat. Unitatea Mobilă de Expansiune (opțional) înseamnă economie de spațiu și rază mare de acțiune, prin transformarea noului Armada M300 într-o adevărată stație de luptă: acces la încă o baterie, încă un hard disc, la CD-ROM (sau DVD), unitate de dischetă etc.
Cu noul Armada M300 de partea ta, războiul e ca și câștigat... pentru că se termină înainte de a începe.

intel inside
pentium® III

ntel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

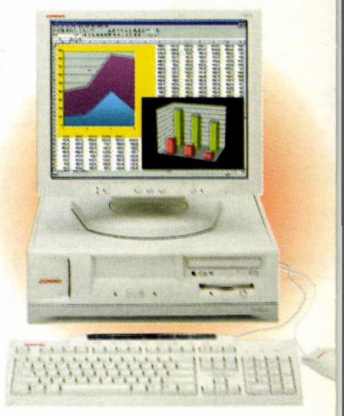
NonStop™

net consulting

Str. Știrbei Vodă 15, București, România
Tel: 223 27 01, 223 26 80; Fax: 312 41 99
E-mail: sales@netconsulting.ro
www.netconsulting.ro

Din păcate, majoritatea consideră că o relație la birou este nepotrivită.

Iar motivul cel mai des invocat este c
ar fi contra-productivă. În realitat
nimic mai fals!
Cu noul Compaq Deskpro EN Sli
Desktop cu procesor Intel® Pentium®
866 MHz poți stabili o relație extre
de productivă încă din prima zi. Pentr
că este un sistem compact, prietenc
și adaptabil, care nu-ți va cre
niciodată vreo problemă.
Deskpro EN Slim Desktop concentrea
tehnologie de vârf într-un spațiu c
20% mai mic decât al oricărui alt P
hard-disc silențios Ultra ATA-100 c
până la 20GB, unitate CD-ROM, 128 M
memorie RAM, adaptor Intel Fa
Ethernet integrat și multe altel
În concluzie, o legătură de durat
inteligentă, va fi oricând acceptată c
restul colegilor ca fiind firească.
Să fie acesta singurul motiv pentru ca
ești privit cu admirație...?



intel inside
pentium® III

Intel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

NonStop™

OMNILOGIC

Șos. București - Ploiești 135
Tel: 303 31 00; Fax: 303 31 64
www.omnilogic.ro

Gazeta de informatică
Revista viitorilor specialiști în tehnologia informației
Decembrie 2000
Vol. 10/8
Lei 15.000

din sumarul numărului:

Interviu
câștigătorii concursului

acm

intel inside™

pentium 4

TEHNO

Intel revine

BABEL

Aplicații multimedia

ENJOY Ce mai spun SURSELE?

PROBLEME de la ACM

Cum să pui puterea informației în slujba ta?

Cum să hrănești mai multe gur deodată? Și cum să le asigur toate cele necesare fără întrerupere?! Non stop?!
Simplu - noua generație de servere Compaq ProLiant ML53 îți sare în ajutor.
Compaq ProLiant ML530 cu procesor Intel® Pentium® II Xeon™ 866 MHz, cu configurați dual-procesor este cel mai rapi server cu arhitectură 2-way di lume. ProLiant ML530 este gândi pentru exploatarea spațiulu interior la maxim: până la 4 G de memorie; 16 bay-uri; 8 slotur PCI; capacitate maximă d depozitare a datelor: 436,8 GB Și asta nu e tot: fiabilitatea ș stabilitatea sistemului sun garantate prin programel proprietare Compaq.

intel inside
pentium® III xeon™

Intel, logotipul Intel Inside, Pentium sunt mărci înregistrate, iar Pentium III Xeon este marcă comercială a Intel Corporation.

COMPAQ

NonStop™

scop® COMPUTERS

Barbu Văcărescu 162
Tel: 231 65 78
Fax: 231 65 96
www.scop.ro

Vol. 10, Nr. 8 /2000

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu (București)*
Redactor șef:
Clara Ionescu (Cluj)
Redactor șef adjunct:
Mihai Scorțaru (Cluj)
Redactor:
Carmen Șușman (Alba Iulia)

Au mai colaborat:
Alexandru Andoni (Chișinău)
Vlad Bazon (Vaslui)
Bogdan Dumitru (Ploiești)
Crina Groșan (Cluj)
Liviu Lalescu (Craiova)
Manuela Mihăescu (Cluj)
Mihai Oltean (Cugir)
Mihai Stroe (București)

Tehnoredactare:
Călin Borz

Coperta și grafica:
Adrian Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihaela Szabo (mszabo@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*

Tel: 01/330.92.81
Fax: 01/330.92.85

Sumar

editorial.. PAG. 3

E-ditorial

actual.. PAG. 5

La Universitatea POLITEHNICA din București
Mihai SCORȚARU

probleme.. PAG. 7

propuse pentru REZOLVARE

Problemele de la ACM

interviu.. PAG. 11

Despre ACM

În dialog cu echipa câștigătoare a concursului ACM

soluții.. PAG. 15

RUBIK
Liviu LALESCU

concurs.. PAG. 21

Rezultatele primei runde

Runda a treia

focus.. PAG. 33

Reprezentarea BINARĂ a NUMERELOR mari - CUM și DE CE
Vlad BAZON

tehno.. PAG. 37

INTEL revine
Mihai SCORȚARU

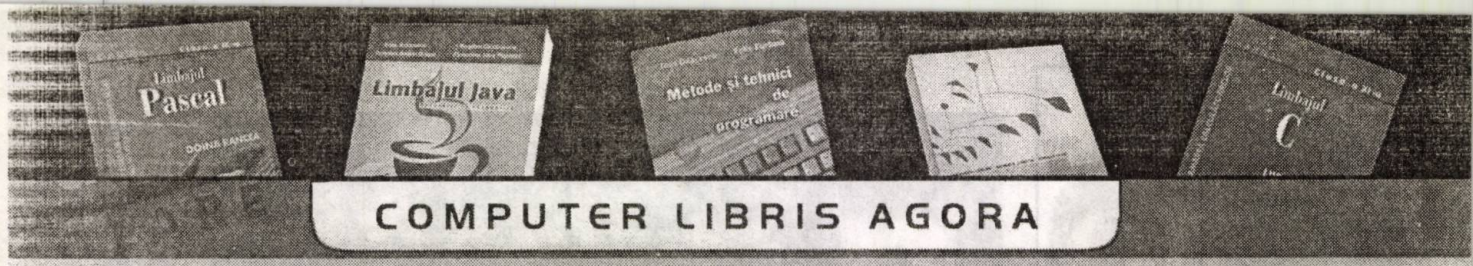
Pentium 4

babel.. PAG. 39

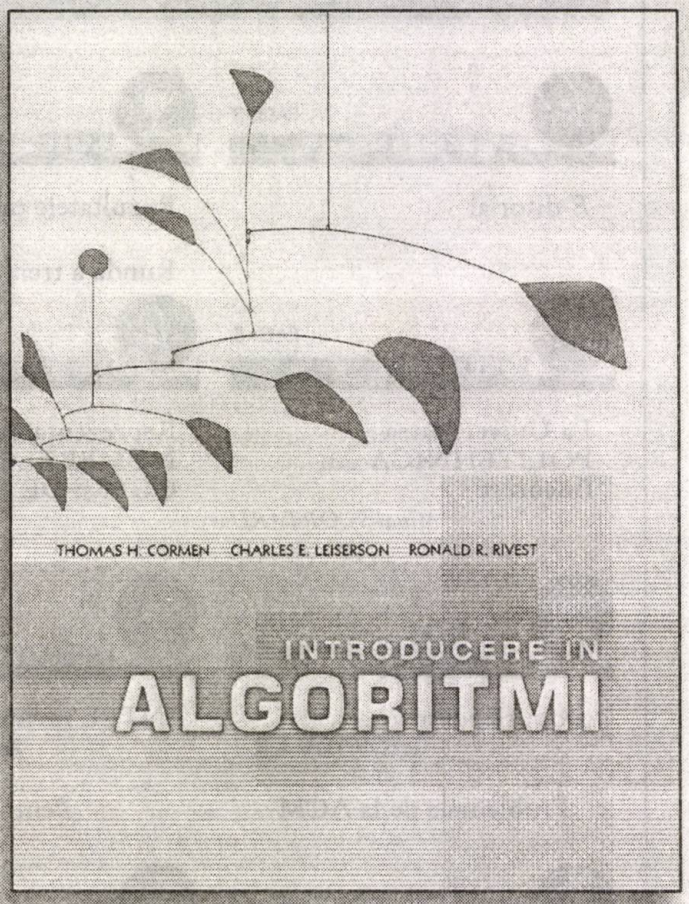
Aplicații MULTIMEDIA în procesul EDUCAȚIONAL
Manuela MIHĂESCU

enjoy.. PAG. 44

Ce ne mai spun SURSELE?
Mihai OLTEAN
Crina GROȘAN



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

E-ditorial

Între 3 și 5 decembrie s-a desfășurat la Sinaia Conferința Internațională "New Modern Technologies and their Impact on Education and Training in the South Eastern Europe", sub egida "Stability Pact for South Eastern Europe, Education and Youth - Enhanced Grax Process - Task Force".

Participanți din întreaga *Europă* au venit la *Sinaia* pentru a-și împărtăși experiența dobândită în domeniu. A fost un prilej deosebit de bun pentru a lega prietenii, parteneriate și de a identifica potențiale proiecte comune în cadrul *Pactului de Stabilitate*.

Într-un moment în care lumea întreagă se întreabă dacă "va intra *România*, în sfârșit în *Uniunea Europeană*, sau nu?", a venit Europa la noi, să discutăm despre ***impactul noilor tehnologii în educație***, despre șansele oferite de o ***economie globală digitală***. Oare de câți ani vorbim de "noi" tehnologii? Și oare câți ani vom mai vorbi? Nu voi încerca să enumăr acele instrumente noi (hard și soft) care în curând vor deveni "mobilier" obișnuit în fiecare unitate economică și în fiecare școală. Am citit și ascultat destule previziuni care, fie s-au îndeplinit cu mult înainte "de termen", fie niciodată. Oare *Internet*-ul a fost prezis de cineva? Azi mulți dintre noi nu am mai putea să ne rezolvăm multe din problemele zilnice fără *Internet*.

Așa se pare că se apropie o eră în care în fața multor noțiuni va apărea litera *e*, precizând că este vorba de varianta electronică. Ce înseamnă *e-Business*, *e-Comerț* sau *e-Learning*? Ați ghicit! Atunci când în procesul de predare/învățare formatul electronic și instrumentele care facilitează folosirea și interpretarea informațiilor stocate în format electronic joacă un rol dominant, vorbim despre *e-Learning*. Este oare România pregătită pentru a asigura accesul tinerilor și nu doar a lor, la acest mod de a învăța?

Puteți să polemizați cu mine. Puteți să insistați în a susține că noțiunea de ***e-Profesor*** este o aberație. Bineînțeles că rolul clasic de a dirija, supraviza procesul de învățare nu va fi înlocuit de nici o mașină electronică. Cel puțin nu prea curând. Dar ce părere aveți despre acel profesor care în clasa lui este înconjurat de calculatoare în rețea, unde accesul la *Internet* este o banalitate și se realizează o navigare rapidă, ieftină și eficientă? Acest profesor va folosi videoproiectorul, va avea la dispoziție o serie de materiale didactice digitale, tutoriale, demo-uri, programe de instruire asistată de calculator, indiferent ce disciplină predă. Sărmanul profesor... Multe va avea de învățat.

Unul din vorbitorii la conferință a insistat asupra fenomenului care poate să ducă la un decalaj mare între țările lumii dacă nu se ține cont de ea. Despre ce este vorba?

Se constată că degeaba se accelerează instruirea în *IT&C*, degeaba se mărește numărul studenților în cadrul instituțiilor de învățământ superior unde se predă informatică și tehnologia informației, degeaba informatica este introdusă ca disciplină de studiat la tot mai multe facultăți, numărul absolvenților de specialitate mereu este mai mic decât numărul de specialiști cerut de piața muncii.

Care este soluția? E simplu: *trebuie să învățăm în același ritm în care se dezvoltă tehnologia.*

Trăim în epoca revoluției informaționale, în epoca informației stocate și prelucrate electronic, așadar este evident că trebuie să folosim și în educație mijloacele pe care *era digitală* ni le pune la dispoziție. Dacă se fac afaceri pe cale electronică (*e-Business* este mai rapid, mai sigur, mai ieftin), se face comerț electronic (pentru ***e-Trade*** nu este nevoie să te deplasezi, poți vedea, întreba, plăti pe cale electronică), atunci de ce nu și ***e-Learning***? Care sunt avantajele? Evident este atractiv, rapid, eficient și ieftin. "Predarea" este exactă, concisă, ilustrată cu imagini și sunet, cu modele și simulări imposibil de realizat într-o lecție clasică la clasă. Permite exersarea, recapitularea, revenirea fără "obosirea" profesorului și în ritmul propriu al elevului/studentului.

Vizitați site-ul destinat discuțiilor pentru *Societatea Informațională în România* (www.eromania.net) și veți afla ce înseamnă ***eRomania*** și ***eEurope***.

Despre eforturile care se depun în *Europa de Sud-Est pentru o dezvoltare democratică și pașnică prin susținerea și coordonarea unor proiecte în domeniul educației* aflați vizitând site-ul www.see-educoop.net.

Toate aceste eforturi guvernamentale nu ar putea fi duse rapid și eficient la finalizare, dacă nu s-ar încheia parteneriate cu firme, cu *NGO*-uri, și dacă ***școala*** nu ar depune și ea eforturi substanțiale.

Un exemplu strălucit îl constituie activitatea depusă de ***Cisco Systems*** în România (www.cisco.com). Programul *Cisco Networking Academy* este un program educațional complex cu peste 5425 de centre în 75 de țări din lume.

În prezent România ocupă locul al șaptelea în lume, fiind înființate 75 de academii locale și regionale.

Înscrieți-vă!

Clara Ionescu

COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

La Universitatea POLITEHNICA din București

Mihai Scorțaru

În luna octombrie s-a desfășurat faza regională a concursului internațional de programare, organizat de ACM și IBM. Ca de obicei, și de data aceasta, concurenții din sud-estul Europei s-au întâlnit la București pentru a încerca să se califice la faza finală.

Despre ACM

ACM este un concurs de programare destinat în exclusivitate studenților. Spre deosebire de majoritatea concursurilor pentru elevi, acesta nu este un concurs individual; universitățile sunt reprezentate de echipe, având cel mult trei membri, iar fiecare echipă are la dispoziție un singur calculator. Este clară intenția organizatorilor de a promova munca în echipă.

Modul în care este stabilit clasamentul final este și el diferit de cel de la concursurile pentru elevi. Primul criteriu este numărul de probleme rezolvate. Dacă mai multe echipe rezolvă același număr de probleme, acestea vor fi departajate pe baza unui al doilea criteriu, și anume punctele de penalizare acumulate.

Modul în care se face corectarea este și el diferit față de concursurile pentru elevi. Pe fiecare stație este instalat un program (PC^2) prin intermediul căruia se realizează legătura dintre concurenți și juriu. Cea mai importantă funcție a acestui program este aceea de trimitere a unei soluții. După trimiterea unei soluții aceasta va fi corectată în cel mai scurt timp posibil și rezultatul corectării este comunicat echipei care a trimis soluția. În momentul în care o soluție a fost acceptată ca fiind corectă se calculează punctele de penalizare corespunzătoare acestei soluții. Numărul de puncte de penalizare este numărul de minute care a trecut de la începerea concursului până la momentul trimiterii soluției corecte, la care se adaugă 20 de puncte de penalizare pentru fiecare soluție incorectă care a fost trimisă pentru problema respectivă.

Pe parcursul desfășurării concursului, mai puțin în ultimele 30 de minute, este afișat un clasament intermediar care prezintă situația în fiecare moment.

Pentru ca toți participanții să cunoască situația la un moment dat, echipele care rezolvă o problemă primesc un balon. Fiecărei probleme îi corespunde un balon de o anumită culoare. Există reguli foarte stricte în ceea ce privește baloanele: spargerea unui balon (obținut de echipa proprie sau de o altă echipă) atrage după sine descalificarea.

O problemă este considerată ca fiind rezolvată numai dacă furnizează răspunsurile corecte pentru toate datele de teste folosite. În cazul în care pentru unul sau mai multe teste răspunsul furnizat de program nu este cel corect, problema este respinsă. Există patru motive pentru care o rezolvare poate fi respinsă: rezultatul furnizat de program nu este cel corect, a apărut o eroare de execuție, s-a depășit limita de timp admisă sau rezultatul este corect, dar ieșirea nu respectă formatul specificat în enunțul problemei.

Un rol foarte important în desfășurarea concursului îl au *runner*-ii. Aceștia sunt cei care comunică rezultatele evaluării, aduc baloanele sau programele pe care concurenții doresc să le listeze.

Pentru a păstra secretul rezultatelor finale, cu 30 de minute înaintea terminării concursului, acceptarea unei rezolvări ca fiind corectă nu mai este confirmată. Totuși, dacă o rezolvare este respinsă, acest lucru este comunicat celor care au trimis rezolvarea respectivă. Rezultatele finale oficiale nu

sunt comunicate decât la festivitatea de premiere care se desfășoară în seara zilei de concurs.

Fiecare universitate poate fi reprezentată de maximum două echipe, iar un student care a participat deja la două faze finale nu se mai poate prezenta la o fază regională.

Pe lângă cei maximum trei studenți care participă la concurs, echipa mai are un antrenor (*coach*).

Dacă o echipă se califică la faza finală, unul dintre membrii echipei poate fi înlocuit de un alt student al aceleiași universități.

ACM 2001

La această ediție a etapei regionale au participat 44 de echipe din țările *Europei de Sud-Est*. Concurenții au avut la dispoziție calculatoare dotate cu procesoare *Intel Pentium III* cu frecvența de 733 MHz, iar sistemul de operare folosit a fost *Microsoft Windows 2000 Professional*.

Concursul a fost câștigat de o echipă a *Universității Politehnica* din *București*, a doua clasată fiind o echipă din *Sofia*, *Bulgaria*. Aceste două echipe vor reprezenta *Europa de Sud-Est* la etapa finală care se va desfășura la *Vancouver*, *Canada* în primăvara anului 2001.

După festivitatea de deschidere a avut loc o sesiune de acomodare. În timpul acestei sesiuni concurenții au avut de rezolvat o problemă și au fost simulate condițiile de concurs.

Enunț

Dându-se un număr întreg n, să se determine suma numerelor întregi pozitive impare care sunt mai mici decât numărul dat n.

Rezolvare

Problema nu este foarte dificilă, o simplă formulă matematică fiind suficientă pentru a o rezolva pentru valori oricât de mari ale lui n. Formula este $1 + 3 + 5 + \ldots + 2n - 1 = n^2$. Deci, problema se rezolva calculând valoarea $[n/2]^2$. Surprinzător a fost faptul că foarte puține echipe au reușit să găsească această formulă. Ea poate fi demonstrată foarte ușor prin inducție matematică.

Pentru $n = 1$, avem $1 = 1^2$, deci formula este adevărată în acest caz.

Considerăm adevărată formula pentru valoarea n și o vom demonstra pentru valoarea $n + 1$. Conform ipotezei de inducție avem:

$$1 + 3 + 5 + \ldots + 2n - 1 = n^2.$$

Dacă adunăm în ambii membri valoarea $2n + 1$ obținem:

$$1 + 3 + 5 + \ldots + 2n - 1 + 2n + 1 = n^2 + 2n + 1 = (n + 1)^2.$$

Am demonstrat corectitudinea formulei și pentru valoarea $n + 1$, deci, conform principiului inducției matematice, formula este adevărată pentru orice valoare pozitivă a lui n.

Concursul

Având în vedere faptul că în timpul sesiunii de acomodare totul a funcționat perfect, nimeni nu credea că vor fi probleme în timpul concursului. Și totuși...

Concursul trebuia să înceapă la ora 10. Concurenții au primit enunțurile problemelor în jurul orei 12. Echipele au început să rezolve problemele, dar, la un moment dat, au observat că nu pot trimite soluții. Unii au aflat că serverul nu mai funcționează, dar acest lucru nu a fost anunțat în mod oficial. Până la urmă s-a anunțat că dificultățile apărute au fost depășite și că totul va reintra în normal. Totuși, după un timp, concurenții au observat din nou că ceva nu este în regulă. Nici de această dată nu a fost anunțat nimic în mod oficial. A fost făcut din nou un anunț potrivit căruia lucrurile au intrat din nou în normalitate. Chiar dacă totul ar fi trebuit să se termine în jurul orei 15, concurenții au părăsit sala de concurs după ora 18. Cu 30 de minute înaintea terminării concursului, singura echipă care rezolvase trei din cele opt probleme a fost cea a *Universității Politehnice* din *Timișoara*. Mai erau câteva echipe care rezolvaseră câte două probleme.

În total au fost opt probleme, două dintre ele nefiind rezolvate de nici una dintre echipe. Timpul de execuție admis a fost de 10 (zece) milisecunde pentru șapte dintre cele opt probleme și "câteva" secunde pentru a opta (*Companie*). Nu s-a știut niciodată ce înseamnă "câteva".

Premierea

Cu mult înainte de anunțurile oficiale se știa faptul că echipa câștigătoare este cea a *Universității Politehnica* din *București*. Din aceste motive, anunțul final nu a mai mirat pe nimeni. Totuși, surprinzător a fost faptul că echipa bucureșteană a rezolvat, până la urmă, cinci din cele opt probleme. Echipa gazdă nu mai câștigase etapa regională a concursului ACM de trei ani.

Concluzii

Această ediție a avut deficiențe din punct de vedere al organizării. Pe lângă numeroasele întârzieri care au apărut, rezultatele finale nu au putut fi ținute secrete așa cum prevede regulamentul.

Redacția *GInfo* felicită echipele câștigătoare și le urează mult succes la etapa finală.

În acest număr puteți citi și interviul pe care ni l-au acordat câștigătorii. Echipa *Universității Politehnica* din *București* a fost formată din *Alexandru Andoni*, *Bogdan Dumitru* și *Mihai Stroe*. Tot în acest număr sunt prezentate enunțurile celor opt probleme de la acest concurs. Rezolvările acestor probleme vor fi publicate în numărul următor.

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa skortzy@xnet.ro.

ACM 2001

Vă prezentăm enunțurile celor opt probleme pe care au trebuit să le rezolve cele 44 de echipe participante la etapa sud-est europeană a concursului studențesc de programare ACM.

P080001: Joc de strategie

Lui Bob îi plac jocurile pe calculator, în special cele de strategie, dar, uneori, el nu poate găsi soluția suficient de repede și, din acest motiv, este foarte trist. Acum el are următoarea problemă. Trebuie să apere un oraș medieval ale cărui străzi formează un arbore. El trebuie să amplaseze în intersecții un număr minim de soldați astfel încât să poată supraveghea toate străzile. Îl puteți ajuta?

Programul dumneavoastră va trebui să determine numărul de minim de soldați care trebuie amplasați pentru un arbore dat.

Fișierul de intrare **A.DAT** conține mai multe seturi de date. Fiecare set de date descrie un arbore astfel:

- numărul de noduri
- descrierea fiecărui nod în formatul următor:

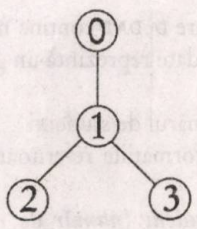
identificator_nod:(*număr_de_străzi*) $identificator\_nod_1$, $identificator\_nod_2$, ..., $identificator\_nod_{număr\_de\_străzi}$

sau

identificator_nod:(0)

Pentru un arbore cu n noduri ($1 \le n \le 1500$) identificatorii nodurilor sunt numere întregi cuprinse între 0 și $n - 1$. Fiecare muchie apare o singură dată în descrierea arborelui.

De exemplu, pentru arborele



soluția este un singur soldat amplasat în nodul 1.

Rezultatul va fi tipărit la ieșirea standard. Pentru fiecare set de date se va tipări, pe câte o linie, un singur număr întreg care va indica numărul minim de soldați.

Exemplu

Intrare

```
3
0:(1) 1
1:(2) 2 3
2:(0)
3:(0)
5
3:(3) 1 4 2
1:(1) 0
2:(0)
0:(0)
4:(0)
```

Ieșire

```
1
2
```

P080002: Companie

O companie plătește diferite sume pentru diferite tipuri de servicii prestate. În fiecare săptămână compania ține evidența numărului total de ore lucrate pentru fiecare tip de servicii și suma totală plătită tuturor angajaților în acea săptămână. În săptămâni diferite pot fi prestate servicii diferite. Remunerația orară pentru un anumit tip de serviciu rămâne neschimbat în cadrul unei companii. Remunerația orară pentru oricare dintre tipurile de servicii este un număr întreg pozitiv, iar raportul dintre remunerația maximă și cea minimă este mai mic decât 6.

Trebuie să scrieți un program care determină remunerațiile orare pentru fiecare dintre tipurile de servicii folosind datele colectate într-o anumită perioadă de timp (una sau mai multe săptămâni). Numărul tipurilor de servicii este cel mult 200, iar numărul de săptămâni este cel mult 50.

Fișierul de intrare **B.DAT** constă în mai multe seturi de date.

Formatul unui set de date este următorul:

- numărul de linii corespunzător setului de date
- informații pentru una sau mai multe săptămâni

Formatul informațiilor pentru o săptămână este următorul:

- *tip_serviciu*$_1$ *număr_de_ore*$_1$
- *tip_serviciu*$_2$ *număr_de_ore*$_2$
- *tip_serviciu*$_3$ *număr_de_ore*$_3$
- ...
- . *suma_totală* (punctul marchează sfârșitul informațiilor referitoare la o anumită săptămână)

Tipurile de servicii sunt reprezentate prin șiruri de maximum 20 de caractere.

Numărul de ore este un număr întreg pozitiv mai mic decât 100.000.

Suma totală este un număr întreg pozitiv mai mic decât 2.000.000.000.

Programul va trebui să tipărească la ieșirea standard (pentru fiecare dintre tipurile de servicii din setul de date) o linie care va conține unul dintre tipurile de servicii și remunerația orară corespunzătoare.

Ieșirea pentru un set de date se va încheia cu o linie care va conține un punct.

Dacă nu poate fi determinată o singură remunerație orară pentru unul dintre tipurile de servicii, se va tipări `Incomplete data`, iar dacă nu poate fi determinată o remunerație orară care să fie număr întreg, se va tipări `Inconsistent data`.

Exemplu

Intrare

```
5
job1  6
job2  5
job8  4
job10 3
.     100
13
job3  1
job2  2
.     40
job1  3
job2  2
.     100
job1  1
job3  2
job2  3
.     100
job1  1
job2  5
.     80
```

Ieșire

```
Incomplete data
.
job3  20
jjob2  10
job1  30
.
```

P080003: Multiplu

Scrieți un program care, dându-se un număr întreg N cuprins între 0 și 4999 (inclusiv), și M cifre zecimale distincte $x_1, x_2, ..., x_M$ (cel puțin una), determină cel mai mic multiplu strict pozitiv al numărului N care nu conține nici o altă cifră cu excepția celor date.

Fișierul de intrare `C.DAT` conține mai multe seturi de date, separate printr-o linie vidă, fiecare set de date având următorul format:

- pe prima linie numărul N
- pe a doua linie numărul M
- pe următoarele M linii cifrele $x_1, x_2, ..., x_M$.

Pentru fiecare set de date programul va tipări la ieșirea standard, pe o singură linie, multiplul determinat. Dacă nu există un astfel de multiplu, se va tipări 0.

Exemplu

Intrare

```
22
3
7
0
1

2
1
1
```

Ieșire

```
110
0
```

P080004: Fete și băieți

În al doilea an universitar cineva a început un studiu al relațiilor romantice dintre studenți. Relația *implicat romantic* este definită între o fată și un băiat.

Din motive specifice studiului, este necesar să se determine mulțimea maximă care satisface condiția: în această mulțime nu există nici o pereche de studenți care să fie implicați romantic.

Rezultatul va fi numărul de studenți dintr-o astfel de mulțime.

Fișierul de intrare `D.DAT` conține mai multe seturi de date. Fiecare set de date reprezintă un grup de studiu descris astfel:

- pe prima linie numărul de studenți
- pe câte o linie informațiile referitoare la un student în formatul:

  identificator_student: (*număr_de_relații_romantice*) *identificator_student*$_1$ *identificator_student*$_2$ ...

sau

  identificator_student: (0).

Pentru un grup de n studenți aceștia sunt identificați prin numere de la 0 la n - 1.

Pentru fiecare set de date, programul va tipări la ieșirea standard o linie care va conține rezultatul.

Exemplu

Intrare

```
7
0: (3) 4 5 6
1: (2) 4 6
2: (0)
3: (0)
4: (2) 0 1
5: (1) 0
6: (2) 0 1
3
0: (2) 1 2
1: (1) 0
2: (1) 0
```

Ieșire

```
5
2
```

P080005: Derivare simbolică

Scrieți un program care efectuează derivarea simbolică `f'(x)=df(x)/dx` asupra unei funcții date `f(x)`. Funcția `f(x)` este definită printr-o expresie care poate conține următorii operatori: + (adunare), - (scădere), * (înmulțire), / (împărțire) și `ln` (logartitm natural).

În afara operatorilor mai poate apărea variabila `x` precum și diferite constante numerice. Expresia poate conține și expresii imbricate arbitrare care sunt cuprinse între paranteze. Expresia este dată în formă infixată. Un exemplu de astfel de expresie este

```
(2*ln(x+1.7)-x*x)/((-7)+3.2*x*x)+(x+3*x)*x.
```

Constantele numerice au forma `d.d` și pot avea și un semn (+ sau -). Numărul de cifre ale părților întreagă și fracționară este arbitrar. Se garantează corectitudinea expresiei date (nu pot apărea erori de sintaxă).

Expresia furnizată la ieșire va fi în formă infixată. Nu trebuie să fie optimizată pentru a putea fi citită mai ușor. Aceasta înseamnă că poate conține redundanțe cum ar fi `0*x`, `1*x`, `0+x` etc. Derivarea trebuie efectuată folosind următoarele reguli:

- Operatorii * și / au prioritate mai mare decât operatorii + și -. Parantezele pot schimba prioritățile.
- Operatorii +, -, * și / sunt asociativi la stânga, adică pot fi grupați de la stânga la dreapta:
  a*b*c=(a*b)*c, a/b/c=(a/b)/c, a/b*c=(a/b)*c etc.
- Regulile de derivare sunt:
  `(a+b)'=a'+b'`
  `(a-b)'=a'-b'`
  `(a*b)'=(a'*b+a*b')`
  `(a/b)'=(a'*b-a*b')/b^2` (folosiți `b^2` și nu `b*b`)
  `ln(a)'=(a')/(a)`
  `x'=1`
  `const'=0`
- Folosiți parantezele doar în cazurile în care apar în regulile anterioare. Nu efectuați optimizări de tipul `0*a=0`, `1*a=a` etc.

Fișierul de intrare **`E.DAT`** va conține câte o definiție a unei funcții `f(x)` pe fiecare linie. Liniile nu conțin blancuri. Ieșirea va consta în linii care vor conține derivatele simbolice corespunzătoare `f'=df(x)/dx`, câte o linie pentru fiecare funcție. Se garantează faptul că șirurile corespunzătoare funcțiilor `f(x)` și `f'(x)` au cel mult 100 de caractere.

Exemplu

Intrare

```
x*x/x
-45.78*x/x
-2.45*x*x+ln(x-3)
```

Ieșire

```
((1*x+x*1)*x-x*x*1)/x^2
(0*x-45.78*1)+1
((0*x-2.45*1)*x-2.45*x*1)+(1-0)/(x-3)
```

P080006: Dreptunghiuri

Un specialist în proiectarea VLSI trebuie să decidă dacă există anumite componente care se acoperă una pe alta pentru un proiect dat. O componentă este reprezentată de un dreptunghi. Se presupune că fiecare dreptunghi are laturile paralele cu axele de coordonate, deci un dreptunghi este reprezentat prin coordonatele *x*, *y* minime și maxime.

Scrieți un program care numără dreptunghiurile care sunt integral acoperite de un alt dreptunghi.

Fișierul de intrare **`F.DAT`** constă în descrierea mai multor seturi de dreptunghiuri. Un set constă în numărul de dreptunghiuri și lista dreptunghiurilor date prin coordonatele *x*, *y* minime și maxime, separate prin blancuri. Formatul este

n
$xmin_1$ $xmax_1$ $ymin_1$ $ymax_1$
$xmin_2$ $xmax_2$ $ymin_2$ $ymax_2$
...
$xmin_n$ $xmax_n$ $ymin_n$ $ymax_n$

Rezultatul va fi tipărit la ieșirea standard. Pentru fiecare set de dreptunghiuri se va tipări un număr întreg care va indica numărul dreptunghiurilor acoperite.

Exemplu

Intrare

```
3
100 101 100 101
0 3 0 101
20 40 10 400
4
10 20 10 20
10 20 10 20
10 20 10 20
10 20 10 20
```

Ieșire

```
0
4
```

P080007: Cursuri

Se consideră un grup de N studenți și P cursuri. Fiecare student frecventează zero, unu sau mai multe cursuri. Sarcina dumneavoastră este de a determina dacă este posibilă formarea unui comitet din P studenți care să satisfacă simultan condițiile:

- fiecare student din comitet reprezintă un anumit curs (un student poate reprezenta un curs, dacă frecventează acel curs);
- fiecare curs este reprezentat în comitet.

Fișierul de intrare **G.DAT** conține mai multe seturi de date. Fiecare set de date are următorul format:

$P\ N$
$nr_1\ student_{1,1}\ student_{1,2} \ldots\ student_{1,nr_1}$
$nr_2\ student_{2,1}\ student_{2,2} \ldots\ student_{2,nr_2}$
...
$nr_p\ student_{p,1}\ student_{p,2} \ldots\ student_{p,nr_p}$

Prima linie a fișierului conține numărul seturilor de date. Prima linie a fiecărui set conține două numere P și N separate printr-un singur spațiu: P ($1 \le P \le 100$) reprezintă numărul cursurilor, iar N ($1 \le N \le 300$) reprezintă numărul studenților.

Fiecare dintre următoarele P linii descrie un curs. Descrierea unui curs începe cu numărul nr_i ($0 \le nr_i \le N$) al studenților care frecventează cursul respectiv. Urmează un spațiu și apoi cei nr_i studenți care frecventează cursul respectiv.

Studenții sunt identificați cu numere întregi cuprinse între 1 și N.

Nu există linii vide între seturile de date, iar datele de intrare sunt corecte.

Programul va tipări la ieșirea standard câte o linie care va conține șirul YES, dacă se poate forma un comitet respectând condițiile date și NO, în caz contrar. La începutul liniilor nu trebuie să apară spații.

Exemplu

Intrare

```
2
3 3
3 1 2 3
2 1 2
1 1
3 3
2 1 3
2 1 3
1 1
```

Ieșire

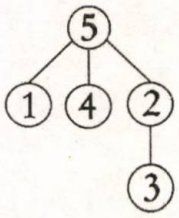
```
YES
NO
```

P080008: Strămoș comun

Scrieți un program care, dându-se un arbore cu rădăcină și o listă de perechi de noduri, determină pentru o pereche (u, v) cel mai apropiat strămoș comun al nodurilor u și v.

Cel mai apropiat strămoș comun a două noduri u și v este nodul w care este strămoș al ambelor noduri și are o adâncime cât mai mare în arbore.

Un nod poate fi propriul său strămoș (de exemplu, pentru arborele din figură, strămoșii nodului 2 sunt nodurile 2 și 5).



Fișierul de intrare **H.DAT** conține mai multe seturi de date cu descrierea arborelui și lista de perechi de noduri. Arborele este descris astfel:

nr_de_noduri
nod: (*număr_de_succesori*) $succesor_1\ succesor_2 \ldots\ succesor_n$
...

Nodurile sunt reprezentate prin numere de la 1 la N, unde N este numărul nodurilor din arbore. Descrierea arborelui este urmată de o listă de perechi de noduri în următorul format:

nr_de_perechi
$(u\ v)\ (x\ y) \ldots$

Caracterele albe pot fi folosite arbitrar în cadrul fișierului de intrare și au rol de separatori.

Pentru fiecare strămoș comun se va tipări strămoșul urmat de numărul de perechi pentru care acesta este cel mai apropiat strămoș comun.

Rezultatul va fi tipărit la ieșirea standard pe linii separate, în ordinea crescătoare a nodurilor, în formatul *strămoș:număr_perechi*.

Exemplu

Intrare

```
5
   5:(3) 1 4 2
   1:(0)
   4:(0)
   2:(1) 3
   3:(0)
   6
(1 5) (1 4) (4 2)
      (2 3)
(1 3) (4 3)
```

Ieșire

```
2:1
5:5
```

Despre ACM

Mihai Scorțaru

Faza regională a concursului ACM a fost câștigată de a doua echipă a Universității Politehnica din București. Echipa a fost formată din Alexandru Andoni, Bogdan Dumitru și Mihai Stroe. Antrenorul echipei a fost domnul profesor Valeriu Iorga. Am adresat câștigătorilor câteva întrebări referitoare la acest concurs și nu numai. Le mulțumim celor trei pentru amabilitatea cu care au răspuns la aceste întrebări.

Pentru început permiteți-mi să vă felicit pentru câștigarea etapei regionale a concursului ACM. Cum vă simțiți în postura de învingători?

Anul trecut am obținut locul IV la același concurs, loc care nu a fost suficient pentru a ne asigura calificarea. Acest lucru nu ne-a descurajat deloc; dimpotrivă, ne-am ambiționat și mai mult, ținta noastră fiind participarea la *Finală* și obținerea unui loc cât mai bun acolo. Am muncit mult pentru această *Regională*, am făcut primul mare pas spre atingerea obiectivului nostru, acum am început să ne gândim la *Finală*, la ce trebuie să facem acolo și să ne pregătim pentru asta.

Ce părere aveți despre modul în care a fost organizat acest concurs?

A devenit deja o tradiție ca organizarea fazei pentru *Europa de Sud-Est* a concursului *ACM* să cadă în sarcina *Politehnicii București*. De-a lungul anilor, organizarea a devenit din ce în ce mai bună. Astfel, calculatoarele folosite au fost de ultimă generație (PIII la 733 Mhz), iar problemele tehnice apărute în timpul concursului au fost rezolvate rapid de cei de la *System Support* și de la *Scientific Committee* (subliniem acest lucru, prin comparație, de exemplu, cu organizarea finalei *ACM* din 1999 unde a existat o problemă cu teste greșite, dar acest fapt nu s-a remediat decât după concurs).

Cum vi s-au părut cele opt probleme propuse spre rezolvare?

La prima vedere, unele probleme păreau simple și ușor de abordat. De fapt, ele necesitau folosirea unor algoritmi foarte performanți, dublați de implementări extrem de eficiente. Limita de timp impusă pentru fiecare test era extrem de restrictivă, astfel încât sursele trebuiau optimizate "la sânge". Am primit multe mesaje *Time Limit Exceeded*, dar am reușit, în final, să aducem programele la forma cerută. În plus, am putea observa că doar șase dintre cele opt probleme au fost "atacate" (și trebuie să ținem seama că echipele participante la această regională sunt dintre cele mai puternice). Dintre cele șase, doar patru au putut fi rezolvate și de alte echipe decât a noastră.

Știu că toți ați participat în timpul liceului la diferite concursuri naționale și internaționale de informatică. Spuneți-mi care au fost cele mai importante rezultate pe care le-ați obținut în această perioadă?

Alexandru Andoni: Am obținut premiul I la toate olimpiadele naționale la care am participat, din 1995 până în 1999. Am câștigat o medalie de bronz la *Olimpiada Internațională de Informatică* în 1997 și două medalii de argint la același concurs, în 1998 și 1999. În 1999 am participat și la *Olimpiada Internațională de Matematică*.

Dumitru Bogdan: Am obținut pe rând premiul III, premiul II și apoi de două ori premiul I la olimpiadele naționale de informatica. Anul trecut am făcut parte din lotul național și am obținut medalia de argint la *Olimpiada de Informatică a Europei Centrale* din 1999 și medalia de aur la *Olimpiada Internațională de Informatică* din același an.

Mihai Stroe: La faza națională am obținut premiul II în 1995, apoi de trei ori premiul I: în 1996, 1997 și 1998. În 1997 și 1998 am fost component al lotului național și am reprezentat România la două *Olimpiade Internaționale de Informatică*, obținând medalia de bronz în 1997 și medalia de aur cu punctaj maxim în 1998.

În 1997 am obținut și o medalie de bronz la *Olimpiada Balcanică de Informatică*. Aș mai putea spune că acum doi ani am participat la faza finală a concursului *ACM* împreună cu *Valentin Gheorghiță* și *Ovidiu Gheorghioiu*.

Care sunt, în opinia voastră, principalele diferențe dintre aceste concursuri și ACM?

Acest concurs, deși propune spre rezolvare aproximativ același gen de probleme ca și concursurile din liceu, prezintă multe diferențe față de acestea. În primul rând, aici sunt trei coechipieri și un singur calculator, deci greșelile unuia singur afectează întreaga echipă; nimeni nu mai este pe cont propriu. În al doilea rând, sursele sunt evaluate în timpul concursului. Se trimite un mesaj foarte scurt (*OK* sau tipul erorii apărute), iar echipa poate reface programul respectiv. Acest sistem se apropie de sistemul folosit de firmele producătoare de soft: se scrie un program, se testează de către cât mai mulți utilizatori, iar pe baza răspunsurilor lor programul este refăcut până la obținerea unei versiuni cât mai bune, care este lansată pe piață (echivalentul răspunsului *OK* de la *ACM*). La *ACM* se pot folosi listing-uri, porțiuni de cod deja scrise, ceea ce mărește mult viteza de lucru și permite echipei să se concentreze asupra elementelor de noutate ale problemelor. În principiu, la o problemă se lucrează în doi, cel care scrie sursa fiind asistat de altul, care poate observa din timp greșelile. Astfel, un program are șanse mult mai mari de a rula perfect "din prima", sau dacă nu, timpul care trebuie pierdut pentru depanare este mult redus, ceea ce contează mult.

Ați fost reprezentanții universității care a organizat concursul. A constituit acest lucru un avantaj?

Faptul că am concurat "la noi acasă" ne-a adus atât avantaje cât și dezavantaje. Din punct de vedere al probei în sine nu am avut nici un avantaj; cel mult, putem spune că eram mai familiarizați cu locul unde a avut loc concursul. Dar, în același timp, participarea la un concurs care se desfășoară într-un oraș străin (într-o țară străină) îți poate oferi multe satisfacții, de care noi nu am avut parte.

În majoritatea universităților din România nu se acordă nici o importanță concursului ACM. Studenții afirmă că nimeni nu se ocupă de pregătirea lor. Este diferită situația în cazul vostru?

Cel puțin la noi lucrurile stau altfel. În primul rând datorăm mult antrenorului nostru, domnul *Valeriu Iorga*, *adviser*-ului nostru, domnul *Nicolae Cupcea*, ca și domnului *Nicolae Țăpuș*, director al concursului *ACM* pentru *Europa de Sud-Est*. Pregătirea oferită de profesorii din *Politehnică* este la un înalt nivel și aceasta a contribuit mult la succesul nostru. Foarte important este și faptul că majoritatea olimpicilor naționali și internaționali se înscriu la *Politehnică*. Astfel, ne-am putut întâlni și forma această echipă și am avut o concurență puternică, ceea ce, de asemenea, a contat mult. În *Politehnică* se dă o importanță deosebită acestui concurs, poate și datorită faptului că aproape în fiecare an, una dintre cele 60 de echipe participante la *Finală* este reprezentanta acestei universități.

Credeți că experiența acumulată în timpul liceului a reprezentat un avantaj?

Categoric, experiența câștigată în liceu constituie un plus. Când am ajuns la facultate, eram deja obișnuiți cu programarea și concursurile de programare; știam foarte bine ce înseamnă un program competitiv, o complexitate minimă, un set de teste, ca și multe alte noțiuni esențiale. Majoritatea celor care participă la *Finala* acestui concurs au fost olimpici în timpul liceului.

Toți v-ați înscris la Concursul de Programare Agora organizat de Gazeta de Informatica. Care au fost motivele pentru care ați ales să participați la acest concurs?

Tuturor ne plac concursurile de informatică și considerăm orice concurs serios o provocare. *CPA* este cu siguranță un concurs adevărat, așa că ne-am înscris fără a sta pe gânduri. De asemenea, acest concurs reprezintă un bun antrenament pentru *Finala ACM* din martie.

Mihai Stroe: Am câștigat ediția de anul trecut a concursului organizat de *GInfo*, așa că îmi doresc mult o reeditare a acestei performante.

Alexandru Andoni și *Bogdan Dumitru:* Vrem să participăm alături de *Mihai* la etapa finală a concursului *CPA* și, de ce nu, vom încerca păstrarea titlului în cadrul echipei. Dar, până atunci trebuie să ne prezentăm bine la etapele desfășurate prin e-mail.

Credeți că GInfo poate contribui, în vreun fel, la pregătirea elevilor și studenților pentru concursurile de informatică?

După părerea noastră deja o face (și încă foarte bine) prin articole teoretice, probleme propuse și rezolvate, prezentarea unor tehnici de programare noi etc.

În cazul vostru a contribuit GInfo în vreun fel?

Prin tot ceea ce am enumerat mai sus, ca și prin acest concurs, aflat la a doua ediție, *GInfo* ne ajută mult în pregătirea noastră. Și noi am publicat articole în această revistă, singura de acest gen din țară (poate chiar din întreaga lume), prin care am încercat să împărtășim din experiența noastră, pentru a ajuta olimpicii de liceu în efortul lor de a ajunge cât mai sus în domeniul care îi interesează.

Cine credeți că sunt cei care au contribuit cel mai mult la pregătirea voastră?

Pregătirea noastră a început de ani buni, alături de profesorii noștri din

liceu. Apoi am format această echipă, antrenată de domnul *Valeriu Iorga*, care ne-a adus seturi dificile de probleme și am putut simula condițiile de concurs în antrenamentele noastre. Acestea, împreună cu experiența anterioară, ne-au permis să formăm o adevărată echipă, cu o tactică foarte bună de concurs, fapt decisiv în concursul de acum câteva săptămâni.

La ACM, chiar dacă rezultatele finale ar fi trebuit să fie secrete până la festivitatea de premiere, aproape toți participanții știau că voi sunteți câștigătorii chiar dacă, cu o jumătate de oră înaintea terminării concursului, nu aveați decât două probleme rezolvate. Cum explicați acest lucru?

Știam că rezolvasem cinci probleme deoarece la ultimele *submit*-uri nu primisem nici un mesaj de eroare și, statistic, eram aproape siguri de victorie (chiar dacă o altă echipă reușea același lucru, i-am fi învins la timp, deoarece am fost prima echipă cu două probleme rezolvate). Dacă vreți, simțeam victoria și aceasta se citea, credem, pe fețele noastre.

În ultimele 30 de minute ați rezolvat trei probleme. Cum ați reușit acest lucru?

Știam că aveam trei programe care mai aveau nevoie doar de câteva optimizări pentru a fi perfecte; am reușit să le terminăm la timp și să le trimitem. Am reușit să demonstrăm matematic că nu se puteau obține algoritmi care să aibă ordine de complexitate mai bune decât ale noastre și, deoarece nu puteam să greșim la implementarea algoritmilor, am trimis programele, ne-am ținut degetele încrucișate și, din fericire, toate cele trei programe au funcționat corect pentru datele de test folosite de juriu.

Felicitări încă o dată pentru victorie și mult succes la faza finală. Doriți să transmiteți un mesaj cititorilor GInfo?

Da, am vrea să-i încurajăm să participe la concursuri, la orice concurs la care li se oferă șansa să ia parte. Românii sunt în top peste tot, atât la nivel de liceu cât și la nivel universitar. Este mult de lucru, dar un concurs de acest gen îți dă sentimentul că poți face aproape orice îți propui și, nu în ultimul rând, îți deschide multe porți; este o rubrică extrem de atrăgătoare în orice CV.

Rezultatele FINALE

1. Universitatea POLITEHNICA București

Membrii: Alexandru Andoni, Bogdan Dumitru, Mihai Stroe
Coach: Valeriu Iorga

5 probleme rezolvate, 924 puncte de penalizare

2. Universitatea ST. KLIMENT OHRIDSKI Sofia

Membrii: Panayot Dobrikov, Preslav Nakov, Nickolay Nickolov
Coach: Stanislav Ovcharov

3 probleme rezolvate, 515 puncte de penalizare

3. Universitatea POLITEHNICA Timișoara

Membrii: Adrian Ion, Marcel Bărbulescu, Sergiu Ștefanov
Coach: Ioan Stoianov

3 probleme rezolvate, 610 puncte de penalizare

4. Universitatea Belgrad (I)
3 probleme rezolvate, 650 puncte de penalizare

5. Universitatea Transilvania, Brașov
3 probleme rezolvate, 702 puncte de penalizare

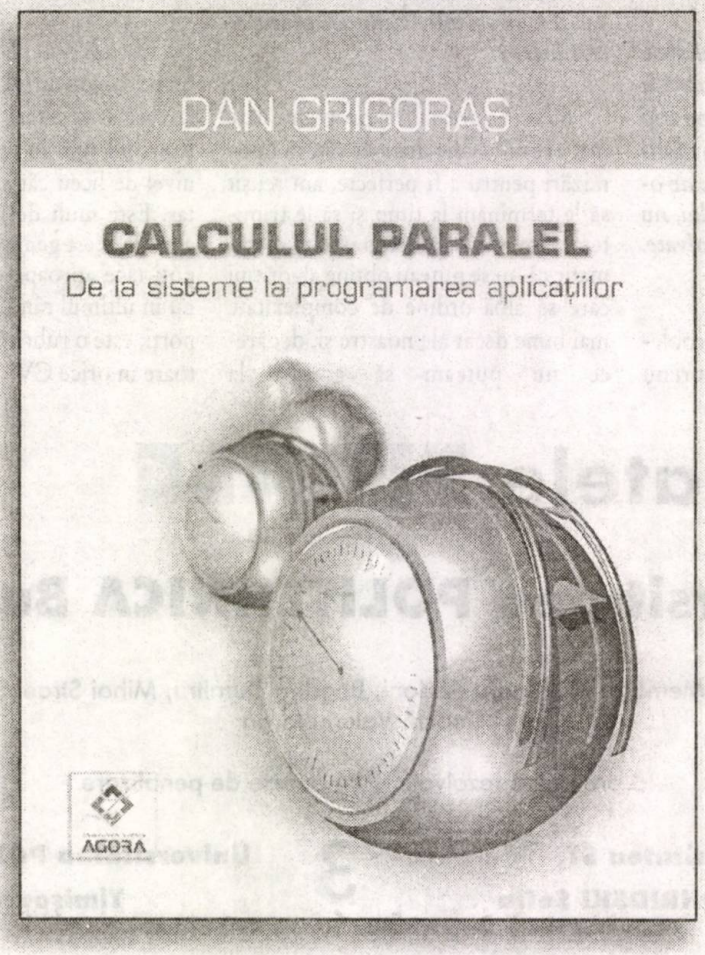
6. Universitatea A.I.Cuza, Iași
2 probleme rezolvate, 192 puncte de penalizare

7. Universitatea Belgrad (II)
2 probleme rezolvate, 261 puncte de penalizare

8. Universitatea Babeș-Bolyai Cluj
2 probleme rezolvate, 286 puncte de penalizare

9. Universitatea POLITEHNICA București (II)
2 probleme rezolvate, 299 puncte de penalizare

COMPUTER LIBRIS AGORA



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate, dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performață și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

RUBIK

Liviu Lalescu

Etapa a șasea a concursului Bursele Agora de anul trecut propunea spre rezolvare o problemă pentru care nu exista o soluție acceptabilă în acel moment. Problema Rubik, devenită celebră după ce a apărut la diverse concursuri de programare din România, a ambiționat mulți programatori doritori să o rezolve. În acest articol vă prezentăm o posibilă rezolvare a acestei probleme.

Enunț

Se dă o grilă de dimensiuni $m \times n$ (m linii, n coloane), care conține în fiecare celulă un număr natural din intervalul [1, 1000].

O operație constă în a aduna valoarea k ($-1000 \le k \le 1000$) la numerele conținute într-o anumită celulă și în toate celulele adiacente pe verticală sau orizontală (nu și pe diagonală).

Determinați o secvență de operații care duce la obținerea aceleiași valori v ($0 \le v \le 1000$) în toate cele $m \times n$ celule ale grilei.

Datele de intrare sunt citite din fișierul **RUBIK.IN** care conține pe prima linie numerele m și n, separate printr-un singur spațiu, iar pe următoarele m linii câte n numere. Al j-lea număr de pe linia i a fișierului corespunde numărului din a j-a celulă de pe linia i a grilei. Numerele de pe aceeași linie vor fi separate prin spații.

Datele de ieșire vor fi scrise în fișierul **RUBIK.OUT**. Prima linie va conține valoarea v, iar a doua numărul x al operațiilor care trebuie efectuate. Următoarele x linii vor conține câte trei numere. Primele două vor indica linia și coloana (în această ordine) celulei alese, iar al treilea valoarea k, care va fi adunată la numărul din celulă și din celulele adiacente pe orizontală sau verticală.

Programul va trebui să furnizeze o secvență corectă de operații, nu neapărat cu număr minim de pași.

Soluție

Problema se reduce la rezolvarea unui sistem de $m \times n$ ecuații cu $m \times n$ necunoscute, astfel încât soluția să conțină numai valori întregi. După rezolvarea acestui sistem obținem o matrice `sol` care va conține valorile care trebuie adunate în cadrul operației corespunzătoare celulei respective.

Construirea sistemului

Este clar că dificultatea constă în a determina prima linie sau prima coloană a matricei `sol`, după aceea, prin parcurgere de sus în jos sau de la stânga la dreapta putem determina în mod unic celelalte valori ale matricei `sol` (adică sistemul poate fi simplificat la n sau m ecuații). Soluția prezentată în continuare determină prima linie a matricei `sol`, deci vom avea un sistem de n ecuații cu n necunoscute.

După ce am determinat valorile de pe prima linie a matricei `sol`, restul valorilor matricei pot fi determinate astfel:

```
// se determina prima linie a matricei sol
for(j=1;j<=n;j++)
  sol[0][j]=0;
for(i=2;i<=m;i++)
  for(j=1;j<=n;j++)
    sol[i][j]=valfin-a[i-1][j]-sol[i-2][j]-
      sol[i-1][j-1]-sol[i-1][j]-sol[i-1][j+1];
```

Am notat cu `valfin` valoarea care se va obține în toate celulele grilei.

Fie x un vector care reprezintă prima linie a matricei `sol`, iar y un vector care reprezintă ultima linie. Căutăm o relație generală între x și y de tipul $Ax = y$. Pornind de la o matrice unitate și urmărind modul de influențare al valorilor `sol` de pe rândul $i + 2$ de către valorile `sol` de pe rândul i și $i + 1$, vom calcula iterativ matricea A. Esențial este faptul că `valfin` și matricea inițială nu intervin în calculul matricei A. Această idee duce la o rezolvare asimptotic mai rapidă.

Pentru vectorul y există anumite constrângeri, deoarece suntem pe ultimul rând și nu mai putem adăuga valori

pe linia $m + 1$. Aceste constrângeri sunt de tipul `y[j]= b[j]`, unde `b[j]` depinde de `valfin` și de matricea inițială.

Rezolvarea sistemului

Așadar am ajuns la sistemul $Ax = b$, pentru care trebuie să obținem soluții întregi. Există numeroase metode de rezolvare a unui sistem de ecuații liniare, toate având complexitatea $O(n^3)$.

Programul trebuie să rezolve, pentru toate valorile finale posibile (de la 0 la 1000), un sistem de n ecuații cu n necunoscute. Rezultă tot o complexitate $O(n^3)$, dar constanta care se ascunde în spatele notației O este foarte mare.

În [1] sunt prezentate metode rapide de rezolvare a astfel de sisteme. Sunt prezentați doi algoritmi: descompunerea *LUP* a matricei sistemului (*LUP_descompunere* cu ordinul de complexitate $O(n^3)$), și rezolvarea prin substituție înainte sau înapoi (*LUP_soluție* cu ordinul de complexitate $O(n^2)$).

Tot în [1] apare o remarcă pe cât de evidentă pe atât de utilă: *descompunerea matricei se poate executa o singură dată, apoi pentru diferite valori ale vectorului coloană b vom aplica procedura LUP_soluție.*

Această observație se poate aplica la problema de față: vom determina întâi descompunerea matricei sistemului, *AP=LU* (matricea *A* nu depinde de `valfin`).

Pentru fiecare valoare `valfin` posibilă vom determina vectorul coloană *b* (*b* depinde de `valfin`). Apelăm apoi subrutina *LUP_soluție*, care va determina vectorul coloană *x* și, dacă fiecare element al vectorului *x* este un întreg, se va returna valoarea 1 (***avem soluție***). În caz contrar se returnează valoare 0 (***nu există soluție pentru valoarea finală curentă***). Dacă obținem o soluție, o vom tipări în fișierul de ieșire și vom opri execuția programului.

Analiza complexității

Programul poate fi rezumat la următoarele apeluri

```
determină A                 //O(m*n^2)
LUP_descompunere            //O(n^3)
pentru valfin de la 0 la 1000
  determină b               //O(m*n)
  dacă LUP_soluție=1        //O(n^2)
    calculează și afișează soluția
    oprește execuția programului
  sfârșit dacă
sfârșit pentru
afișează "Nu există soluție"
```

Rezultă o complexitate finală având ordinul $O(n^3 + valoare\_finală\_maximă \cdot n^2)$. Timpul de execuție devine acum acceptabil.

Erori de propagare

O altă problemă care apare constă în erorile de propagare. Aceste erori nu afectează rezultatul pentru valori mici, dar duc la rezultate eronate pentru dimensiuni mai mari ale matricei.

Dacă se folosește tipul **`long double`** din C++, pentru matrice de dimensiuni 35×35, se obține o eroare de propagare destul de mică, astfel încât primele două cifre semnificative de după virgulă sunt corecte (de exemplu se obține valoarea `11.993454` care poate fi considerată `12`). Pentru dimensiuni mai mari, programul se execută repede dar incorect. Pentru a rezolva corect problema, trebuie simulate operații pe numere cu mai multe cifre semnificative. Am apelat la biblioteca *GNU-Arithmetic Algorithms*, distribuită gratuit și care oferă funcții excelent implementate. Am folosit o precizie de 192 de biți pentru mantisă (pentru tipul **`long double`** se folosesc 64 de biți).

Memoria insuficientă

Mai există o problemă în *Borland C++ 3.1*, legată de alocarea de memorie, și de aceea trebuie folosit un compilator mai performant în privința alocării memoriei (*Borland C++Builder*, *GNU gcc*). Programul folosește aproximativ $2 \cdot 100 \cdot 100 \cdot 12 = 240.000$ de octeți de memorie alocați static și $2 \cdot 100 \cdot 100 \cdot 24 = 480.000$ de octeți alocați dinamic pentru mantise. Pentru setul de date cel mai defavorabil programul rulează în mai puțin de un minut pe un calculator dotat cu procesor *AMD Duron* la 600 MHz, în sistemul de operare *Linux*. Pentru *Borland C++Builder 4.0* sub *Windows* timpul de execuție este puțin mai mare.

Îmbunătățiri posibile

În primul rând am putea verifica dacă $n > m$. În acest caz, o transpunere a matricei ar duce la micșorarea ordinului de complexitate, de la n la m. O altă îmbunătățire caută să minimizeze timpul de execuție astfel: se calculează media aritmetică a numerelor din matrice și se pornește ciclul de la acea valoare, testând apoi pentru valori apropiate. Aceasta abordare duce la soluții mult mai rapide pentru seturi de date generate aleator. Folosind acest artificiu programul s-a încadrat în timp pentru toate testele folosite la concursul *Bursele Agora* organizat de *GInfo*.

Fișierul sursă

Am ales să vă prezentăm în paginile revistei versiunea scrisă pentru sistemul de operare *Linux* deoarece este mai rapidă și ocupă mai puțin spațiu. Această sursă, precum și cea scrisă folosind mediul *Borland C++Builder 4.0*, sunt disponibile pentru *download* la **www.ginfo.ro**.

Listing RUBIK.C

```
#include "gmp.h"
#include "gmp-impl.h"
#include "mp.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
```

```
#define MAXM 100
#define MAXN 100
#define eps 0.001

int m,n,a[MAXM+2][MAXN+2],
                sol[MAXM+2][MAXN+2],valfin;
mpf_t rez[MAXN+2],b[MAXN+2],b1[MAXN+2],
 b2[MAXN+2],y[MAXN+2],row1[MAXN+2][MAXN+2],
                    row2[MAXN+2][MAXN+2];

int isinteger(mpf_t x){
  double y=mpf_get_d(x);
  y=fabs((double)(y-(int)y));
  return (y<eps||1-y<eps);
}

int integer(mpf_t x){
  double y=mpf_get_d(x);
  if(y>0) return (int)(y+0.5);
  else return (int)(y-0.5);
}

int LUPdesc(mpf_t a[MAXN+2][MAXN+2],
                    int pi[MAXN+2],int n){
//returneaza rangul matricei caracteristice
  int i,j,k,r;
  mpf_t p,tt;
  double ttt;
  for(i=1;i<=n;i++)
    pi[i]=i;
  mpf_init(p),mpf_init(tt);
  for(k=1;k<=n;k++){
    mpf_set_si(p,0);
    for(i=k;i<=n;i++){
      mpf_set(tt,a[i][k]);
      mpf_abs(tt,tt);
      if(mpf_cmp(tt,p)>0){
        mpf_set(p,tt);
        r=i;
      }
    }
    ttt=mpf_get_d(p);
    if(ttt<0.000001){
      mpf_clear(p),mpf_clear(tt);
      return k-1;
      // Matrice singulara presupunem ca
      // avem sistem nedeterminat
    }
    j=pi[k];
    pi[k]=pi[r]; pi[r]=j;
    for(i=1;i<=n;i++){
      mpf_set(p,a[k][i]);
      mpf_set(a[k][i],a[r][i]);
      mpf_set(a[r][i],p);
    }
    for(i=k+1;i<=n;i++){
      mpf_div(a[i][k],a[i][k],a[k][k]);
      for(j=k+1;j<=n;j++){
        mpf_mul(tt,a[i][k],a[k][j]);
        mpf_sub(a[i][j],a[i][j],tt);
      }
    }
  }
  mpf_clear(p),mpf_clear(tt);
  return n;
}

int LUPsol(mpf_t a[MAXN+2][MAXN+2],
           mpf_t b[MAXN+2],int pi[MAXN+2],
                    int n,mpf_t x[MAXN+2]){
// returneaza 0 daca apar numere fractionare
// returneaza 1 daca este OK
  int i,j;
  pf_t suma,tt;
  mpf_init(suma);
  mpf_init(tt);
  for(i=1;i<=n;i++){
    mpf_set_si(suma,0);
    for(j=1;j<i;j++){
      mpf_mul(tt,a[i][j],y[j]);
      mpf_add(suma,suma,tt);
                    // suma+=a[i][j]*y[j];
    }
    mpf_sub(y[i],b[pi[i]],suma);
                    // y[i]=b[pi[i]]-suma;
  }
  for(i=n;i>=1;i--){
    mpf_set_si(suma,0);
    for(j=i+1;j<=n;j++){
      mpf_mul(tt,a[i][j],x[j]);
      mpf_add(suma,suma,tt);
                    // suma+=a[i][j]*x[j];
    }
    mpf_sub(tt,y[i],suma);
    mpf_div(x[i],tt,a[i][i]);
                // x[i]=(y[i]-suma)/a[i][i];
    if(!isinteger(x[i])){
      mpf_clear(suma),mpf_clear(tt);
      return 0;
    }
  }
  mpf_clear(suma),mpf_clear(tt);
  return 1;
}

void read(void){
  long i,j;
  FILE *fi=fopen("RUBIK.IN","rt");
  fscanf(fi,"%d%d",&m,&n);
  memset(a,0,sizeof(a));
```

```
  for(i=1;i<=m;i++)
    for(j=1;j<=n;j++)
      fscanf(fi,"%d",&(a[i][j]));
  fclose(fi);
}

void solve(){
// calculeaza prima linie a solutiei
// sistem de n ecuatii cu n necunoscute
// care sunt cantitatile care se adauga la
// celulele de sus sistemul este:
// [A][x]=[b], unde [A]=row2,[b]=b si
// [x]=rez in prima faza, calculam
// [P][A]=[L][U]
  int i,j,k,nn;
  int pi[MAXN+2];
  mpf_t tt;
  // aloca numerele
  mpf_init(tt);
  for(i=0;i<=n+1;i++)
    for(j=0;j<=n+1;j++){
      mpf_init(row1[i][j]);
      mpf_init(row2[i][j]);
    }
  for(i=0;i<=n+1;i++){
    mpf_init(b1[i]),mpf_init(b2[i]);
    mpf_init(b[i]),mpf_init(rez[i]);
  }
  for(i=0;i<=n+1;i++)
    mpf_init(y[i]);
  for(i=1;i<=n;i++)
    mpf_set_si(row2[i][i],1);
  for(k=2;k<=m+1;k++){
    for(j=1;j<=n;j++)
      for(i=1;i<=n;i++){
        mpf_add(row1[j][i],
                row2[j-1][i],row1[j][i]);
        mpf_add(row1[j][i],
                row2[j][i],row1[j][i]);
        mpf_add(row1[j][i],
                row2[j+1][i],row1[j][i]);
        mpf_neg(row1[j][i],row1[j][i]);
        // row1[j][i]=-row1[j][i]-
        // row2[j-1][i]-row2[j][i]-
        // row2[j+1][i];
      }
    // schimba randurile intre ele
    for(i=0;i<=n+1;i++)
      for(j=0;j<=n+1;j++){
        mpf_set(tt,row2[i][j]);
        mpf_set(row2[i][j],row1[i][j]);
        mpf_set(row1[i][j],tt);
      }
  }
  nn=LUPdesc(row2,pi,n);
```

```
// pentru valfin de la 0 la 1000 incercam
// sa obtinem un sistem cu rezultate
// intregi
for(valfin=0;valfin<=1000;valfin++){
  for(i=0;i<=n+1;i++)
    mpf_set_si(b1[i],0),
                mpf_set_si(b2[i],0);
  for(k=2;k<=m+1;k++){
    for(j=1;j<=n;j++){
      mpf_set_si(tt,valfin);
      mpf_sub(tt,tt,b1[j]);
      mpf_sub(tt,tt,b2[j-1]);
      mpf_sub(tt,tt,b2[j]);
      mpf_sub(tt,tt,b2[j+1]);
      mpf_sub_ui(tt,tt,a[k-1][j]);
      mpf_set(b1[j],tt);
      //b1[j]=valfin-b1[j]-b2[j-1]-
              b2[j]-b2[j+1]-a[k-1][j];
    }
    // schimba randurile intre ele
    for(i=0;i<=n+1;i++){
      mpf_set(tt,b1[i]);
      mpf_set(b1[i],b2[i]);
      mpf_set(b2[i],tt);
    }
  }
  for(i=0;i<=n+1;i++)
    mpf_set(b[i],b2[i]);
  for(i=1;i<=n;i++)
    mpf_neg(b[i],b[i]);
  if(LUPsol(row2,b,pi,nn,rez))
    break;
}
assert(valfin!=1001);
// calculeaza toata matricea
// matricea solutiei o construim in sol
memset(sol,0,sizeof(sol));
for(i=1;i<=nn;i++)
  sol[1][i]=integer(rez[i]);
for(;i<=n;i++)
  sol[1][i]=0; //valoare arbitrara
for(i=2;i<=m;i++)
  for(j=1;j<=n;j++)
    sol[i][j]=valfin-a[i-1][j]-
              sol[i-2][j]-sol[i-1][j-1]-
              sol[i-1][j]-sol[i-1][j+1];
// eliberarea memoriei
mpf_clear(tt);
for(i=0;i<=n+1;i++)
  for(j=0;j<=n+1;j++){
    mpf_clear(row1[i][j]);
    mpf_clear(row2[i][j]);
  }
for(i=0;i<=n+1;i++){
  mpf_clear(b1[i]);
```

```
    mpf_clear(b2[i]);
    mpf_clear(b[i]);
    mpf_clear(rez[i]);
  }
  for(i=0;i<=n+1;i++)
    mpf_clear(y[i]);
}

void write(void){
  FILE *fo=fopen("RUBIK.OUT","wt");
  int i,j,nmoves=0,k;
  for(i=1;i<=m;i++)
    for(j=1;j<=n;j++)
      if(sol[i][j])
        nmoves++;
  fprintf(fo,"%d\n%d\n",valfin,nmoves);
  for(i=1;i<=m;i++)
    for(j=1;j<=n;j++)
      if(sol[i][j])
        fprintf(fo,"%d %d %d\n",i,j,
                                  sol[i][j]);
  fclose(fo);
}

void main(void){
  mpf_set_default_prec(256);
  read();
  solve();
  write();
}
```

Instrucțiuni

În primul rând va trebui să instalați biblioteca `gmp` din arhiva `gmp-3.0.tar.gz` (**www.gnu.org** pachetul *Arithmetic Algorithms*). Apoi se compilează programul **`RUBIK.C`** prin linia de comandă `gcc -g rubik.c -lgmp -lm`.

Bibliografie

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest, *Introducere în algoritmi*, Editura Computer Libris Agora, 2000.

[2] Biblioteca *GNU-Arithmetic Algorithms*, www.gnu.org.

Liviu Lalescu este student în anul III la Facultatea de Automatică, Calculatoare și Electronică din Craiova. Poate fi contactat prin e-mail la adresa **liviulalescu@mailsurf.com**.

Notă

Programul RUBIK a fost testat în redacție și a funcționat corect pentru toate seturile de date folosite. Totuși redacția nu își asumă nici o responsabilitate pentru funcționarea imprevizibilă a programului.

news...

Opera 5.0

Până acum programul de navigare Opera și-a câștigat adepți datorită dimensiunii, vitezei, respectării standardelor actuale și faptului că este disponibil pentru o varietate foarte mare de platforme.

Principalul dezavantaj al acestui program îl reprezenta faptul că, pentru a-l folosi, navigatorii erau nevoiți, cel puțin teoretic, să achiziționeze o licență de utilizare. Situația este diferită în cazul produselor Internet Explorer și Netscape Navigator care sunt distribuite gratuit. Începând cu versiunea Opera 5.0 și acest program de navigare poate fi folosit fără a mai plăti pentru licență.

În acest moment cei doi combatanți principali în războiul procesoarelor încearcă să-și integreze programul de navigare în celelalte produse oferite. De exemplu, Netscape și Microsoft construiesc portale care să fie compatibile cu propriile programe de navigare. Aceste portale intră în competiție cu portalurile independente, iar Opera își propune să fie programul de navigare cel mai potrivit pentru independenți.

Chiar înainte de a fi gratuit, Opera era "a treia forță" în războiul browser-elor. Mulți dintre fanii acestui program de navigare consideră că Opera 5.0 va reuși să se impună pe piață și va fi o alternativă viabilă la Internet Explorer și Netscape Navigator.

Totuși, versiunea gratuită a programului Opera 5.0 vă "obligă" să aveți permanent o reclamă în fereastra de navigare. Banner-ul este încărcat în fiecare săptămână și este salvat pe discul local pentru a fi afișat timp de șapte zile. Există și posibilitatea de a achiziționa o versiune a produsului care nu afișează reclame, dar aceasta costă 39$.

Programul de navigare Opera 5.0 suportă criptarea pe 128 de biți, precum și standardele TLS 1.0, SSL 2, SSL 3, CSS 1, CSS 2, HTML 4.01, HTTP 1.1, ECMAScript, JavaScript 1.3 etc. De asemenea, este în stadiu experimental suportul pentru DOM și WAP/WML.

Opera 5.0 poate fi descărcat gratuit de la adresa www.opera.com/download.

COMPUTER LIBRIS AGORA

"Cum aș putea să justific existența acestei cărți? Cum aș putea spre exemplu să justfic cele 100 de pagini despre programarea dinamică, acum când această metodă nu mai figurează în programele școlare? Nu știu. Sau, cum aș putea să justific capitolul despre algoritmi genetici, când pe plan mondial abia de câțiva ani a fost dovedită utilitatea și eficiența lor? Din nou nu știu! Dar rămâne bănuiala: a scrie algoritmi înseamnă a crea. Și nu poți crea fără să gândești."

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

CONCURSUL de programare AGORA

Prima rundă a Concursului de Programare Agora s-a încheiat. În acest număr vă vom prezenta soluțiile oficiale ale celor trei probleme, precum și clasamentul de la data închiderii ediției. Acest clasament este neoficial, el putând suferi modificări.

Așa cum ne-am așteptat și această ediție a concursului de programare organizat de *Gazeta de Informatică*, s-a bucurat de o largă participare. Suprinzător a fost numărul foarte mare de participanți din afara României. Alături de concurenții români s-au înscris participanți din alte 29 de țări ale lumii: *Africa de Sud*, *Brazilia*, *Bulgaria*, *Canada*, *Cehia*, *Chile*, *Croația*, *Estonia*, *Franța*, *Germania*, *Grecia*, *India*, *Indonezia*, *Iran*, *Iugoslavia*, *Letonia*, *Malayesia*, *Marea Britanie*, *Moldova*, *Olanda*, *Polonia*, *Slovacia*, *Slovenia*, *Spania*, *Sri Lanka*, *Statele Unite ale Americii*, *Thailanda*, *Ungaria* și *Vietnam*.

Până la sfârșitul lunii noiembrie s-au înscris peste 300 de participanți, aproape 100 dintre ei fiind de peste hotare.

Ne face plăcere să anunțăm faptul că *CPA* a devenit un concurs de programare apreciat și recomandat de *Secretariatul Olimpiadei Internaționale de Informatică*.

Prima rundă

Cele trei probleme de la prima rundă au fost foarte simple, dovadă fiind faptul că au existat nu mai puțin decât 28 de concurenți care au obținut punctajul maxim (300 puncte).

Toate cele trei probleme necesitau găsirea unei formule simple pentru obținerea rezultatului. Totuși, datorită valorilor mari care se obțineau trebuiau simulate eficient operațiile cu numere mari.

Deoarece pentru primele două probleme nu erau decât 2457 de posibilități pentru intrare, ieșirile corespunzătoare puteau fi generate printr-un procedeu oarecare, urmând ca apoi să fie scris un program care folosea "*metoda constantelor*" pentru a afișa soluția. Chiar și pentru a treia problemă, la care numărul posibil al combinațiilor pentru intrare era de aproximativ 50.000, putea fi folosită aceeași metodă.

Totuși, pentru a determina valorile, trebuie practic găsiți aceeași algoritmi (implementările nu ar fi trebuit să fie neapărat la fel de eficiente) ca și la rezolvarea "clasică"; ca urmare "metoda constantelor" a fost aleasă de un număr relativ mic de participanți.

Erori

Se pare că o mulțime de participanți nu au citit cu atenție regulamentul concursului. Astfel, am avut programe *Pascal* care foloseau unit-ul `Crt`, programe care nu citeau datele de intrare din fișierele specificate sau scriau datele de ieșire în alte fișiere; s-a folosit și tipul `Comp` fără a include directiva de compilare `{$N+}`. Toate aceste scăpări au dus la pierderea punctelor corespunzătoare unor rezolvări care, de fapt, au fost corecte.

Au existat și concurenți care nu au construit arhiva trimisă așa cum era indicat în regulament. Datorită faptului că era vorba de prima rundă am trecut cu vederea aceste greșeli, dar reconstruirea arhivelor incorecte a întârziat semnificativ procesul de evaluare. Vă rugăm insistent să citiți cu atenție regulamentul concursului și, mai ales, să îl respectați. Cea mai frecventă eroare de acest gen a fost necrearea directorului care conține sursele trimise. Aceste scăpări nu vor mai fi tolerate în viitor.

Cele mai amuzante erori au fost cele cauzate de editarea incorectă a fișierului **`RUNDA1.BAT`** sau **`ROUND1.BAT`**. Am specificat foarte clar faptul că sursele trebuie compilate folosind comenzi care încep cu `bpc` sau `bcc`.

Concluzii

Se pare că CPA este un concurs interesant pentru o mulțime de tineri din întreaga lume. Vă promitem că vom încerca să organizăm cât mai bine acest concurs astfel încât el să se ridice la nivelul așteptărilor voastre și să-și atingă adevăratul scop, și anume acela de a vă ajuta să vă pregătiți pentru concursurile "adevărate" care urmează.

Dacă aveți sugestii sau comentarii referitoare la acest concurs, nu ezitați să ne scrieți.

CLASAMENT - CPA Top 100

1.	*Cristian Alexandrescu, Botoșani*	300
1.	*Andras Csaba, Oradea*	300
1.	*Ciprian Baicu, Dr. Tn. Severin*	300
1.	*Denis Bogdănaș, Iași*	300
1.	*Bors Vencel, Oradea*	300
1.	*Ciprian Cană, Vaslui*	300
1.	*Adrian Cârcu, Bistrița*	300
1.	*Arcadie Crăcan, Iași*	300
1.	*Daniel Crișan, Râmnicu Vâlcea*	300
1.	*George Drumea, București*	300
1.	*Daniel Dumitran, București*	300
1.	*Bogdan Dumitru, Ploiești*	300
1.	*Claudiu Gruia, București*	300
1.	*Ioana-Maria Ileană, Alba Iulia*	300
1.	*Cătălin Kesa, Bușteni*	300
1.	*Liviu Lalescu, Craiova*	300
1.	*Cosmin Negruseri, Bistrița*	300
1.	*Sorin Oțescu, Brașov*	300
1.	*Gabriel Pârvan, Râmnicu Sărat*	300
1.	*Flaviu Pașca, Bistrița*	300
1.	*Mihai Pătrașcu, Craiova*	300
1.	*Mihai Popa, București*	300
1.	*M.A. Sahari Ghahsareh, Iran*	300
1.	*Raluca Sauciuc, București*	300
1.	*Bogdan Stan, Câmpina*	300
1.	*Radu Ștefan, Brașov*	300
1.	*Mihai Stroe, București*	300
29.	Vlad Vâlceanu, Arad	295
30.	Venkitasubramaniam Muthuramakrishnan, India	294
31.	Mugurel Ionuț Andreica, București	293
32.	Aurelian Ghiță, Buzău	288
33.	Alexandra-Elena Constantin, Ploiești	287
33.	Vlad Dascălu, Bacău	287
33.	Dan Ghinea, București	287
33.	Vincent Groenhuis, Olanda	287
33.	Liviu Păunescu, Constanța	287
38.	Horia Ciurdar, Timișoara	283
39.	Dan Popovici, Arad	275
40.	Cosmin Răianu, Constanța	274
41.	Preslav Nakov, Bulgaria	266
42.	Narayanan Arvind, India	264
43.	Petko Minkov, Bulgaria	241
44.	Cristian Băicoianu, Ploiești	230
45.	Andrei Csibi, Cluj-Napoca	220
46.	Răzvan Mușăloiu-Elefteri, București	216
46.	Raluca Mușăloiu-Elefteri, București	216
46.	Edison Nica, Iași	216
49.	Bogdan Nicolae, Sibiu	206
50.	Dao The Phuong, Vietnam	203
50.	Victor Vernescu, Constanța	203
52.	Victor Asavei, Drobeta Turnu Severin	200
52.	Damir Korencic, Croația	200
52.	Codruț-Lucian Lazăr, Câmpeni	200
52.	Mihai-Vlad Pantiș, Cluj-Napoca	200
56.	Radu Dondera, București	187
56.	Artur Kuczapski, Oradea	187
56.	Mojca Miklavec, Slovenia	187
56.	Andrei Tătăranu, Buzău	187
60.	Mihai Gojol, Cluj-Napoca	183
61.	Andrei Adler, București	179
62.	Lucian Burja, Blaj	176
62.	Paul Marinescu, Buzău	176
64.	Andrei David, Râmnicu Vâlcea	172
65.	Mihai Hărănguș, Cluj-Napoca	158
66.	Patcas Csaba, Oradea	146
67.	Andrei Markovits, Satu Mare	141
68.	Adrian Cearnău, București	139
69.	Bogdan-Nicolae Șanta, Cluj-Napoca	138
70.	Alina Fesnic, Cluj-Napoca	135
71.	Cristian Bojinovici, București	129
71.	Silviu-Gabriel Udrea, Târgu Jiu	129
73.	Andrei Giurgiu, București	128
74.	Rusu Paul, Cluj-Napoca	125
75.	Bogdan-Milovan Piloga, Timișoara	122
76.	Ionel-Cristian Iană, București	115
77.	Bogdan Lucaciu, Drobeta Turnu Severin	109
77.	Emilian Miron, Bacău	109
79.	Adrian Bălan, Târgu Jiu	106
80.	Bogdan Mărincaș, Cluj-Napoca	103
81.	Nadina Busuioc, Constanța	100
81.	Perini Arpad, Sfântu Gheorghe	100
81.	Bogdan-Vlad Țira, Cluj-Napoca	100
84.	Valentin Bisa, Lugoj	99
85.	Adrian Bucur, Cluj-Napoca	87
85.	Daniel Comșa, Crăciunelu de Jos	87
85.	Victor Grădinescu, București	87
88.	Octavian Pop, Blaj	84
89.	Daniel Bolohan, Iași	83
90.	Alexandru-Emilian Susu, București	82
91.	Aurel-Mihai Fulger, Constanța	81
92.	Cristian Toth, Lugoj	80
93.	Răzvan Dinu, Ploiești	77
94.	Claudiu-Octavian Mureșan, Arad	71
95.	Horea Coroiu, Cluj-Napoca	67
96.	Marius Cosma, București	61
96.	Andrei-Liviu Dumbrava, Iași	61
98.	Claudiu Ionesi, Bacău	60
99.	Cosmin Rădoi, Timișoara	57
100.	Georgiana Aspru, Râmnicu Vâlcea	54

Runda 1, problema 1

Triangularizare

Această problemă a fost rezolvată corect de către 49 dintre cei care au participat la prima rundă a *Concursului de Programare Agora*. Chiar dacă problema nu este foarte cunoscută în această formă, ea se reduce la a determina numărul de șiruri care pot avea $2n$ elemente și care satisfac criteriile:

- Exact n elemente au valoarea 1 și exact n elemente au valoarea 0.
- Orice prefix al acestor șiruri conține mai multe elemente care au valoarea 1 decât elemente care au valoarea 0; prin prefix înțelegem un subșir care conține primele elemente ale unui șir.

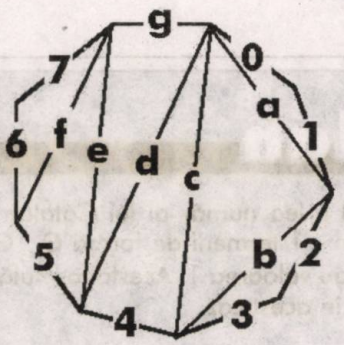
O altă variantă echivalentă a problemei este următoarea: *Să se determine numărul șirurilor de n perechi de paranteze care se deschid și se închid corect*. Aceasta este cea mai cunoscută formă; există și alte enunțuri echivalente ale aceleiași probleme.

Vom demonstra acum echivalența dintre o triangularizare a unui poligon cu n laturi și determinarea unui șir de n - 2 perechi de paranteze care se deschid și se închid corect.

Dându-se o triangularizare oarecare a poligonului, considerăm una dintre laturi ca fiind baza triangulației. Celelalte laturi sunt etichetate cu numere cuprinse între 0 și n - 4. Construim șirul de paranteze așa cum se descrie în continuare.

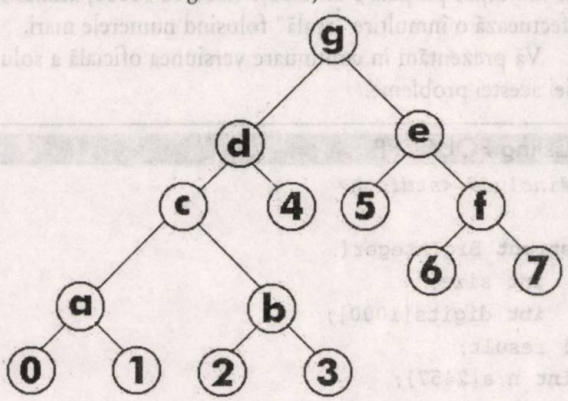
Baza și diagonalele pot fi etichetate dacă fac parte dintr-un triunghi care are celelalte două laturi etichetate. În acest caz, eticheta ei va conține o paranteză dreaptă deschisă, eticheta primei laturi, eticheta celei de-a doua laturi și o paranteză dreaptă închisă.

Să considerăm următoarea triangulație a unui poligon cu 9 laturi:



La primul pas vom eticheta diagonala a cu `[01]`, diagonala b cu `[23]` și diagonala f cu `[67]`. La al doilea pas diagonala c este etichetată cu `[[01][23]]`, iar diagonala e este etichetată cu `[5[67]]`. La al treilea pas etichetăm diagonala d cu `[[01][23]4]`, iar la ultimul baza este etichetată cu `[[[01][23]4][5[67]]]`.

Această etichetare poate fi considerată ca fiind descrierea unui arbore binar în care numerele reprezintă frunzele, iar parantezele descriu modul în care sunt realizate legăturile. Astfel, fiecărei triunghiulariziări îi va corespunde un arbore binar unic. Pentru triangularizarea anterioară avem arborele din figura de mai jos.



Vom eticheta cu 0 muchiile spre fiii drepți și cu 1 muchiile spre fiii stângi. Efectuând apoi o parcurgere a arborelui obținem un șir de numere 1 și 0, care satisface cele două criterii amintite anterior. Înlocuind cifrele 1 cu o paranteză deschisă și cifrele 0 cu o paranteză închisă, obținem o parantezare. Astfel am demonstrat că fiecărei triangulariziări a unui poligon cu n laturi îi corespunde o parantezare cu n - 2 perechi de paranteze.

Se știe că numărul de parantezări cu n perechi de paranteze este numărul lui *Catalan* care are valoarea

$$\frac{C_{2n}^{n}}{n+1} = \frac{2n!}{n!(n+1)!}.$$

Deoarece numărul triunghiulariziărilor unui poligon cu n laturi corespunde unei parantezări cu n - 2 perechi de paranteze, formula pe care o vom folosi este

$$\frac{C_{2n-4}^{n-2}}{n-1} = \frac{(2n-4)!}{(n-2)!(n-1)!}.$$

Deoarece rezultatul poate avea aproape 1500 cifre trebuie implementate operații cu numere mari.

Vom construi un vector care va conține valorile $n - 1$, $n, n + 1, ..., 2n - 4$. Produsul P al acestor numere reprezintă valoarea

$$\frac{(2n-4)!}{(n-2)!}.$$

Pentru a obține rezultatul final va trebui să împărțim acest produs la $(n - 1)!$. Pentru fiecare număr cuprins între 2 și $n - 1$ vom încerca să împărțim produsul P la numărul respectiv. Să considerăm un număr k. Pentru a împărți P la k vom efectua următoarele operații:

```
cât timp k>1 execută
  d ← cmmdc(k,a_i)
  a_i ← a_i / d
  k ← k / d
  i ← i + 1
sfârșit cât timp
```

După efectuarea tuturor împărțirilor, rezultatul final va fi obținut înmulțind elementele vectorului a. Pentru a micșora timpul de execuție vom lucra în baza 10000. De asemenea, nu vom efectua operații cu numere mari atâta timp cât putem realiza înmulțiri parțiale astfel încât rezultatul să nu depășească valoarea 10000. Dacă printr-o astfel de înmulțire parțială s-ar depăși valoarea 10000, atunci se efectuează o înmulțire "reală" folosind numerele mari.

Vă prezentăm în continuare versiunea oficială a soluției acestei probleme.

Listing POLY.CPP

```
#include <stdio.h>

struct BigInteger{
  int size;
  int digits[1000];
} result;
int n,a[2457];

void ReadData(void){
  FILE *f=fopen("POLY.IN","rt");
  fscanf(f,"%d",&n);
  fclose(f);
}

int cmmdc(int m,int n){
  return (m%n)?cmmdc(n,m%n):n;
}

void divide(int k){
  int i=0,d;
  while (k>1){
    d=cmmdc(k,a[i]),a[i++]/=d,k/=d;
}

void multiply(int k){
  int transp=0,newdigit;
  for (int i=0;i<result.size;i++){
    newdigit=((long)result.digits[i]*k+
                                transp)%10000;
    transp=((long)result.digits[i]*k+
                                transp)/10000;
    result.digits[i]=newdigit;
  }
  while (transp){
    result.digits[result.size++]=
                                transp%10000;
    transp/=10000;
  }
}

void ComputeResult(void){
  for (int i=n-1;i<=2*n-4;i++) a[i-n+1]=i;
  for (i=2;i<n;i++) divide(i);
  result.size=result.digits[0]=1;
  int rez=1;
  for (i=0;i<n-2;i++){
    if (rez>10000/a[i])
      multiply(rez),rez=a[i];
    else rez*=a[i];
  }
  multiply(rez);
}

void WriteResult(){
  FILE *f=fopen("POLY.OUT","wt");
  if (n>2){
    fprintf(f,"%d",
            result.digits[result.size-1]);
    for (int i=result.size-2;i>=0;i--)
      fprintf(f,"%04d",result.digits[i]);
  }
  else fprintf(f,"0");
  fclose(f);
}
```

Numerele lui Catalan

Există mai multe posibilități de a calcula valorile numerelor lui Catalan. În afara formulei folosite în prezentarea soluției oficiale, mai poate fi folosită și relația de recurență potrivit căreia al n-lea număr al lui Catalan C_n este suma a n - 1 termeni de forma $C_i \cdot C_{n-1-i}$, iar C_0 și C_1 au valoarea 1. Acestă formulă nu este eficientă în acest caz.

Au rezolvat corect...

Andrei Adler, Bucureşti
Cristian Alexandrescu, Botoşani
Csaba Andras, Oradea
Victor Asavei, Drobeta Turnu Severin
Cristian Băicoianu, Ploieşti
Ciprian-Nicolae Baicu, Dr. Turnu Severin
Denis Bogdănaş, Iaşi
Vencel Bors, Oradea
Ciprian Cană, Vaslui
Adrian Cârcu, Bistriţa
Adrian Cearnău, Bucureşti
Horia Ciurdar, Timişoara
Alexandra-Elena Constantin, Ploieşti
Arcadie Crăcan, Iaşi
Daniel Crişan, Râmnicu Vâlcea
Vlad Dascălu, Bacău
George Drumea, Bucureşti
Octavian-Daniel Dumitran, Bucureşti
Bogdan Dumitru, Ploieşti
Dan Ghinea, Bucureşti
Aurelian Ghiţă, Buzău
Vincent Groenhuis, Olanda
Claudiu Gruia, Bucureşti
Ioana-Maria Ileană, Alba Iulia
Cătălin Kesa, Buşteni
Damir Korencic, Croaţia
Liviu Lalescu, Craiova
Codruţ-Lucian Lazăr, Câmpeni
Venkitasubramaniam Muthuramakrishnan, India
Preslav Nakov, Bulgaria
Cosmin-Silvestru Negruseri, Bistriţa
Sorin Oţescu, Braşov
Mihai-Vlad Pantiş, Cluj-Napoca
Gabriel Pârvan, Râmnicu Sărat
Flaviu Paşca, Bistriţa
Mihai Pătraşcu, Craiova
Liviu Păunescu, Constanţa
Bogdan-Milovan Piloga, Timişoara
Mihail Popa, Bucureşti
Dan Popovici, Arad
Cosmin Răianu, Constanţa
Mohammad Ali Safari Ghahsareh, Iran
Raluca Sauciuc, Bucureşti
Bogdan Stan, Câmpina
Mihai Stroe, Bucureşti
Radu Ştefan, Braşov
Dao The Phuong, Vietnam
Radu Vatavu, Suceava
Victor Vernescu, Constanţa

Date de test

Test	Valoarea N	Rezultat	Număr de cifre	Punctaj
1	11	4862	4	3
2	37	31162...01262	19	3
3	71	33748...24700	39	3
4	80	73745...35600	44	3
5	133	27647...87614	76	4
6	203	20335...86120	118	4
7	276	11414...38984	162	4
8	354	71670...08200	208	4
9	454	79205...95600	268	5
10	513	21902...39645	304	5
11	519	88157...22300	307	5
12	530	35828...01700	314	5
13	610	42387...33000	362	6
14	734	14517...95360	437	6
15	780	65614...57840	464	6
16	787	10606...57928	469	6
17	872	13607...23200	520	7
18	935	10424...61760	558	7
19	997	20132...28000	595	7
20	1128	12394...01120	674	7

Divizori

Această problemă a fost rezolvată corect de către 46 dintre cei care au participat la prima rundă a *Concursului de Programare Agora*. Metoda care ar fi trebuit folosită este cea a ***programării dinamice***.

Vom începe prin a demonstra un adevăr matematic.

Teoremă

Dacă n este un număr natural și a un șir care conține puterile la care apar factorii primi din descompunerea lui n, atunci n are exact $\prod_{i=0}^{\infty}(a_i+1)$ divizori.

Demonstrație

Demonstrația acestei teoreme este foarte simplă. Numărului n i se asociază șirul a_i al puterilor la care apar factorii primi din descompunerea sa. Astfel fiecărui divizor îi va corespunde un element al produsului cartezian $A_1 \times A_2 \times \ldots \times A_n$, unde $A_i = \{0, 1, \ldots, a_i\}$, iar acest produs cartezian are $\prod_{i=0}^{\infty}(a_i+1)$ elemente.

De fapt, nu trebuie să luăm în considerare toate numerele prime, fiind suficiente doar cele mai mici.

În continuare vom presupune că am ales primele 15 numere prime.

La început determinăm vectorul d al divizorilor numărului k. Considerăm că acest vector are m elemente. Construim o matrice A cu 15 linii și m coloane, A_{ij} având valoarea egală cu cel mai mic număr care are ca factori primi primele numere prime și având d_j divizori. Rezultatul cerut va fi valoarea minimă de pe ultima coloană a matricei.

Primul element al fiecărei linii ($A_{i,1}$) va avea valoarea 1, deoarece 1 este cel mai mic număr cu un singur divizor. Vom avea $A_{0,j} = \infty$ deoarece nu există numere cu i divizori și nici un factor prim.

Elementul $A_{i,j}$ poate fi calculat folosind valorile $A_{i-1,k}$ pentru $k < j$ și d_k este divizor al lui d_j. $A_{i-1,k}$ are d_k divizori, deci o valoare A_{ij} cu d_j divizori poate fi obținută calculând $A_{i-1,k} \cdot p_i^{d_j / d_k + 1}$ unde p_i este al i-lea număr prim. Valoarea A_{ij} va fi aleasă ca fiind minimul acestor valori. Dacă această valoare minimă este mai mare decât $A_{i-1,j}$ atunci A_{ij} va primi valoarea $A_{i-1,j}$.

Deoarece ar trebui să lucrăm cu numere foarte mari, în program determinăm logaritmii naturali ai valorilor A_{ij}, iar în final calculăm valoarea reală a minimului de pe ultima coloană a matricei folosind operații cu numere mari.

Vă prezentăm în continuare versiunea oficială a soluției acestei probleme.

Listing DIV.CPP

```
#include <stdio.h>
#include <math.h>
#include <values.h>

struct BigInteger{
  int size;
  long digits[1000];
} result;

const np=6;
const primes[16]={0,2,3,5,7,11,13,17,19,23,
                  29,31,37,41,43,47};

int put[np+1][301];
int ti[np+1][301];
int tj[np+1][301];
int m,n;
int d[401],pp[401];
double ln[np+1];
double a[np+1][401];

void ReadData(void){
  FILE *f=fopen("DIV.IN","rt");
  fscanf(f,"%d",&n);
  fclose(f);
}

void ComputeDivisors(void){
  m=0;
  for (int i=1;i<=n;i++)
    if(!(n%i))
      d[++m]=i;
}

void ComputeLogarithms(void){
  for (int i=1;i<=np;i++)
    ln[i]=log(primes[i]);
}
```

```
void Recursivity(int i,int j){
  if (i<1) return;
  Recursivity(ti[i][j],tj[i][j]);
  pp[i]=put[i][j];
}

void Solve(void){
  for (int i=2;i<=m;i++)
    a[0][i]=MAXLONG;
  for (i=0;i<=np;i++)
    a[i][1]=0;
  for (i=1;i<=np;i++)
    for (int j=2;j<=m;j++){
      a[i][j]=a[i-1][j];
      ti[i][j]=i-1;
      tj[i][j]=j;
      put[i][j]=0;
      for (int k=1;k<=j-1;k++)
        if (!(d[j]%d[k]))
          if (a[i-1][k]+ln[i]*
                (d[j]/d[k]-1)<a[i][j]){
            a[i][j]=a[i-1][k]+ln[i]*
                (d[j]/d[k]-1);
            put[i][j]=d[j]/d[k]-1;
            ti[i][j]=i-1;
            tj[i][j]=k;
          }
    }
  Recursivity(np,m);
}

void multiply(int k){
  int transp=0;
  int newdigit;
  for (int i=0;i<result.size;i++){
    newdigit=((long)result.digits[i]*k+
                transp)%10000;
    transp=((long)result.digits[i]*k+
                transp)/10000;
    result.digits[i]=newdigit;
  }
  while (transp){
    result.digits[result.size++]=
                transp%10000;
    transp/=10000;
  }
}

void FindSolution(void){
  result.size=result.digits[0]=1;
  int rez=1;
  for (int i=1;i<=np;i++)
    for (int j=1;j<=pp[i];j++)
      if (rez<10000/primes[i])
        rez*=primes[i];
      else
        multiply(rez);
        rez=primes[i];
  multiply(rez);
}

void ComputeResult(void){
  ComputeDivisors();
  ComputeLogarithms();
  Solve();
  FindSolution();
}

void WriteResult(){
  FILE *f=fopen("DIV.OUT","wt");
  fprintf(f,"%d",
          result.digits[result.size-1]);
  for (int i=result.size-2;i>=0;i--)
    fprintf(f,"%04d",result.digits[i]);
  fclose(f);
}

void main(void){
  ReadData();
  ComputeResult();
  WriteResult();
}
```

Remarcă

Pentru această problemă cazul cel mai defavorabil apare în situația în care numărul care trebuie determinat are un număr prim de divizori. Dacă numărul *k* al divizorilor trebuie să fie prim atunci produsul care apare în teorema prezentată poate avea doar un termen, deci cel mai mic număr care va avea *k* divizori trebuie să fie un număr prim ridicat la puterea *k*-1.

Deoarece cel mai mic număr prim este 2, dacă *k* este prim, cel mai mic număr cu *k* divizori va fi 2 ridicat la puterea *k*-1. Datorită faptului că, în cazul acestei probleme, numărul divizorilor este cuprins între 1 și 2457, cazul cel mai defavorabil apare pentru cel mai mare număr prim mai mic sau egal cu 2457. Acest număr este 2447.

Au rezolvat corect...

Cristian Alexandrescu, Botoşani
Csaba Andras, Oradea
Mugurel Ionuţ Andreica, Bucureşti
Cristian Băicoianu, Ploieşti
Ciprian-Nicolae Baicu, Drobeta Turnu Severin
Denis Bogdănaş, Iaşi
Vencel Bors, Oradea
Lucian Burja, Blaj
Ciprian Cană, Vaslui
Adrian Cârcu, Bistriţa
Arcadie Crăcan, Iaşi
Daniel Crişan, Râmnicu Vâlcea
Andrei Csibi, Cluj-Napoca
Andrei David, Râmnicu Vâlcea
George Drumea, Bucureşti
Octavian-Daniel Dumitran, Bucureşti
Bogdan Dumitru, Ploieşti
Claudiu Gruia, Bucureşti
Ionel-Cristian Ianã, Bucureşti
Ioana-Maria Ileană, Alba Iulia
Cătălin Kesa, Buşteni
Damir Korencic, Croaţia
Liviu Lalescu, Craiova
Codruţ-Lucian Lazăr, Câmpeni
Emilian Miron, Bacău
Răzvan Muşăloiu-Elefteri, Bucureşti
Raluca Muşăloiu-Elefteri, Bucureşti
Venkitasubramaniam Muthuramakrishnan, India
Preslav Nakov, Bulgaria
Cosmin-Silvestru Negruseri, Bistriţa
Edison Nica, Iaşi
Bogdan Nicolae, Sibiu
Sorin Oţescu, Braşov
Gabriel Pârvan, Râmnicu Sărat
Flaviu Paşca, Bistriţa
Csaba Patcas, Oradea
Mihai Pătraşcu, Craiova
Arpad Perini, Sfântu Gheorghe
Mihail Popa, Bucureşti
Mohammad Ali Safari Ghahsareh, Iran
Raluca Sauciuc, Bucureşti
Bogdan Stan, Câmpina
Mihai Stroe, Bucureşti
Radu Ştefan, Braşov.
Vlad Vâlceanu, Arad
Radu Vatavu, Suceava

Date de test

Test	Valoarea N	Rezultat	Număr de cifre	Punctaj
1	1	1	1	3
2	20	240	3	3
3	55	82944	5	3
4	64	7560	4	3
5	77	746496	6	4
6	81	44100	5	4
7	147	1166400	7	4
8	208	3870720	7	4
9	277	12141...31136	84	5
10	383	98505...76704	115	5
11	495	101606400	9	5
12	565	42057...27776	36	5
13	636	14186...06240	19	6
14	752	66498...92480	17	6
15	879	71614...60064	89	6
16	880	415134720	9	6
17	1010	51339...77280	33	7
18	1085	48922...60000	15	7
19	1203	23240...40384	122	7
20	1398	31057...68320	72	7

Premiere

Această problemă a fost rezolvată corect de către 60 dintre cei care au participat la prima rundă a *Concursului de Programare Agora*.

Problema se reducea la determinarea numărului de partiții în k clase a unei mulțimi având n elemente.

Aceste numere poartă denumirea de *numerele lui Stirling de speța a II-a* și se notează prin $S(n, k)$. Se observă că pentru $k > n$ avem $S(n, 1)=1$ și $S(n, k) = 0$.

Considerăm cele $S(n - 1, k - 1)$ partiții în $k - 1$ clase ale unei mulțimi având $n - 1$ elemente. Adăugând partiției o nouă clasă formată din al n-lea element obținem $S(n - 1, k - 1)$ partiții în k clase ale unei mulțimi având n elemente.

Considerăm acum cele $S(n - 1, k)$ partiții în k clase ale unei mulțimi având n elemente.

Putem adăuga al n-lea la oricare dintre cele k clase, deci vom avea încă $k \cdot S(n - 1, k)$ partiții în k clase ale unei mulțimi având n elemente.

În concluzie, formula de calcul a *numerelor lui Stirling de speța a II-a* este

$$S(n, k) = S(n - 1, k - 1) + k \cdot S(n - 1, k).$$

Datorită valorilor mari care se obțin, vor trebui simulate operațiile pe numere mari.

Vă prezentăm în continuare versiunea oficială a soluției acestei probleme.

Listing AWARDS.PAS

```
#include <stdio.h>

int a[100][200],n,k;

void main(void){
  FILE *f=fopen("AWARD.IN","rt");
  fscanf(f,"%d%d",&n,&k);
  fclose(f);
  a[1][0]=1;
  for (int i=2;i<=n;i++)
    for (int j=i;j;j--)
      for (int tr=0,l=0;l<200;l++){
        tr=(a[j][l]=a[j][l]*j+
                          a[j-1][l]+tr)/10;
        a[j][l]%=10;
      }
  f=fopen("AWARD.OUT","wt");
  for (i=200;(--i) && !a[k][i];);
  for (;i>=0;i--)
    fprintf(f,"%d",a[k][i]);
  fclose(f);
}
```

Referitor la anunțul general...

Se pare că enunțul acestei probleme nu a fost suficient de clar, dovadă fiind faptul că o mulțime de concurenți au cerut detalii suplimentare cu privire la modul în care au fost obținute cele 25 de posibilități de premiere în cazul exemplului dat. Organizatorii Concursului de Programare Agora au decis că este cazul ca toți participanții să primească aceste informații sumplimentare, motiv pentru care a fost trimis un anunț general cu descrierea celor 25 de modalități. Acest lucru a determinat nemulțumirea unora care nu au fost de acord cu modul în care am procedat, deoarece ei și-au dat seama singuri care sunt cele 25 de configurații. Totuși, textul era relativ ambiguu, motiv pentru care ne cerem scuze! După primirea anunțului general, majoritatea concurenților au înțeles enunțul problemei și au rezolvat-o corect. Au fost însă și concurenți care au trimis rezolvări anterior anunțului general. Majoritatea acestora au rezolvat corect problema, dar au fost și concurenți care au trimis rezolvări incorecte. Unii dintre aceștia au citit apoi anunțul general și au trimis din nou rezolvări la această problemă. Potrivit regulamentului acestui concurs, dacă un concurent trimite mai multe variante de rezolvare a problemelor, numai prima dintre ele este luată în considerare. Din acest motiv, programele celor care au trimis o a doua rezolvare pentru această problemă nu au mai fost evaluate, fiind punctată numai prima soluție trimisă.

Au rezolvat corect...

Cristian Alexandrescu, Botoşani
Csaba Andras, Oradea
Mugurel Ionuţ Andreica, Bucureşti
Victor Asavei, Drobeta Turnu Severin
Ciprian-Nicolae Baicu, Drobeta Turnu Severin
Denis Bogdănaş, Iaşi
Vencel Bors, Oradea
Nadina Busuioc, Constanţa
Ciprian Cană, Vaslui
Adrian Cârcu, Bistriţa
Horia Ciurdar, Timişoara
Alexandra-Elena Constantin, Ploieşti
Arcadie Crăcan, Iaşi
Daniel Crişan, Râmnicu Vâlcea
Andrei Csibi, Cluj-Napoca
Vlad Dascălu, Bacău
Radu Dondera, Bucureşti
George Drumea, Bucureşti
Octavian-Daniel Dumitran, Bucureşti
Bogdan Dumitru, Ploieşti
Alina Feşnic, Cluj-Napoca
Dan Ghinea, Bucureşti
Aurelian Ghiţă, Buzău
Mihai Gojol, Cluj-Napoca
Vincent Groenhuis, Olanda
Claudiu Gruia, Bucureşti
Mihai Hărănguş, Cluj-Napoca
Ioana-Maria Ileană, Alba Iulia
Cătălin Kesa, Buşteni
Artur Kuczapski, Oradea
Liviu Lalescu, Craiova
Paul Marinescu, Buzău
Andrei Markovits, Satu Mare
Bogdan Mărincaş, Cluj-Napoca
Mojca Miklavec, Slovenia
Petko Minkov, Bulgaria
Răzvan Muşăloiu-Elefteri, Bucureşti
Raluca Muşăloiu-Elefteri, Bucureşti
Cosmin-Silvestru Negruseri, Bistriţa
Edison Nica, Iaşi
Bogdan Nicolae, Sibiu
Sorin Oţescu, Braşov
Mihai-Vlad Pantiş, Cluj-Napoca
Gabriel Pârvan, Râmnicu Sărat
Flaviu Paşca, Bistriţa
Mihai Pătraşcu, Craiova
Liviu Păunescu, Constanţa
Mihail Popa, Bucureşti
Dan Popovici, Arad
Cosmin Răianu, Constanţa
Mohammad Ali Safari Ghahsareh, Iran
Bogdan-Nicolae Şanta, Cluj-Napoca
Raluca Sauciuc, Bucureşti
Bogdan Stan, Câmpina
Mihai Stroe, Bucureşti
Radu Ştefan, Braşov
Andrei Tătăranu, Buzău
Bogdan-Vlad Tira, Cluj-Napoca
Vlad Vâlceanu, Arad
Radu Vatavu, Suceava

Date de test

Test	Valoarea N	Valoarea K	Rezultat	Număr de cifre	Punctaj
1	6	2	31	2	3
2	8	2	127	3	3
3	11	10	55	2	3
4	16	10	193754990	9	3
5	17	10	2758334150	10	4
6	22	4	72777..23825	12	4
7	22	2	2097151	7	4
8	27	20	12246...12250	14	4
9	31	26	71327...58156	11	5
10	34	27	56062...85080	15	5
11	43	3	54709...68501	20	5
12	57	37	11693...57842	42	5
13	60	8	37907...08939	50	6
14	66	39	20025...78000	55	6
15	70	55	50605...28000	37	6
16	70	22	32894...43395	73	6
17	71	42	21927...79480	60	7
18	82	28	31137...98800	89	7
19	93	13	62971...30981	94	7
20	98	49	14238...86500	100	7

Termen: 15 Ianuarie 2001!

Runda 3

Vă prezentăm în continuare cele trei probleme de la cea de-a treia rundă a Concursului de Programare Agora. Concurenții care nu s-au înscris până la data de 5 Decembrie 2000 pot participa la concurs începând cu această rundă, dacă trimit mesajul de înscriere până pe data de 10 Ianuarie 2001.

CPAR3P1: Galbeni

Nume fișier sursă: `GOLD.PAS`, `GOLD.C` sau `GOLD.CPP`
Nume fișier de intrare: `GOLD.IN`
Nume fișier de ieșire: `GOLD.OUT`

Descriere: Se dau n ($1 \le n \le 1000$) grămezi de galbeni. Dacă o grămadă A conține a galbeni, o grămadă B conține b galbeni și $b \le a$, atunci din grămada A pot fi mutați b galbeni în grămada B. Efectuând mai multe operații, trebuie să obținem cel mult două grămezi de galbeni (restul de n - 2 grămezi nu vor mai conține nici un galben). Grămezile sunt etichetate cu numere de la 1 la n. Numărul total al galbenilor nu poate depăși valoarea 2.000.000.000.

Date de intrare: Fișierul de intrare va conține pe prima linie numărul n, iar pe a doua linie n numere reprezentând numărul de galbeni din cele n grămezi.

Date de ieșire: Fișierul de ieșire va conține câte o linie pentru fiecare operație efectuată. Fiecare linie va conține două numere; primul va indica grămada din care se iau galbeni, iar al doilea grămada în care se pun galbeni. Numărul de operații efectuate nu trebuie să fie neapărat minim.

Exemplu:

`GOLD.IN`	`GOLD.OUT`
4	3 1
1 2 3 4	1 2
	2 4

Timp de execuție: 1 secundă/test.

CPAR3P2: Fantome

Nume fișier sursă: `GHOSTS.PAS`, `GHOSTS.C` sau `GHOSTS.CPP`
Nume fișier de intrare: `GHOSTS.IN`
Nume fișier de ieșire: `GHOSTS.OUT`

Descriere: Un grup de n ($1 \le n \le 500$) vânători luptă cu n fantome. Vânătorii și fantomele se află în puncte având coordonate reale din intervalul [0, 1000]. Nu există trei puncte coliniare în care să se afle vânători sau fantome. Vânătorii au arme fotonice cu ajutorul cărora pot trimite raze care distrug fantomele. Fiecare vânător va trebui să distrugă câte o fantomă. Armele sunt descărcate în același timp și nu este permisă intersecția a două raze, deoarece efectele nu mai pot fi controlate. Scrieți un program care determină care fantomă trebuie distrusă de fiecare vânător.

Date de intrare: Prima linie a fișierului de intrare conține numărul n. Următoarele n linii conțin coordonatele punctelor în care se află vânătorii. Următoarele n linii conțin coordonatele punctelor în care se află fantomele.

Date de ieșire: Fișierul de ieșire va conține n numere, pe câte o linie, a i-a linie indicând fantoma distrusă de al i-lea vânător. Fantomele sunt etichetate cu numere de la 1 la n.

Exemplu:

`GHOSTS.IN`	`GHOSTS.OUT`
3	2
0 0	1
4 3	3
4 0	
2 1	
2 5	
5 5	

Timp de execuție: 2 secunde/test.

CPAR3P3: Unitate

Nume fișier sursă: `UNIT.PAS`, `UNIT.C` sau `UNIT.CPP`
Nume fișier de intrare: `UNIT.IN`
Nume fișier de ieșire: `UNIT.OUT`

Descriere: Se dă o matrice de dimensiuni $m \times n$ ($1 \le m, n \le 100$) care conține doar elemente din mulțimea `{0,1}`. Fiecare linie și coloană reprezintă scrierea binară a unui număr natural (în cadrul unei linii numărul se citește de la dreapta la stânga, iar în cadrul unei coloane numărul se citește de jos în sus). Singura operație admisă este adunarea binară a unei unități la numărul de pe o linie sau de pe o coloană. Determinați numărul de unități care trebuie adunate pentru a obține o matrice ale cărei elemente au toate valoarea `1`.

Date de intrare: Prima linie a fișierului de intrare conține valorile m și n. Următoarele m linii conțin câte n numere reprezentând valorile elementelor matricei.

Date de ieșire: Fișierul de ieșire va conține numărul operațiilor care trebuie efectuate.

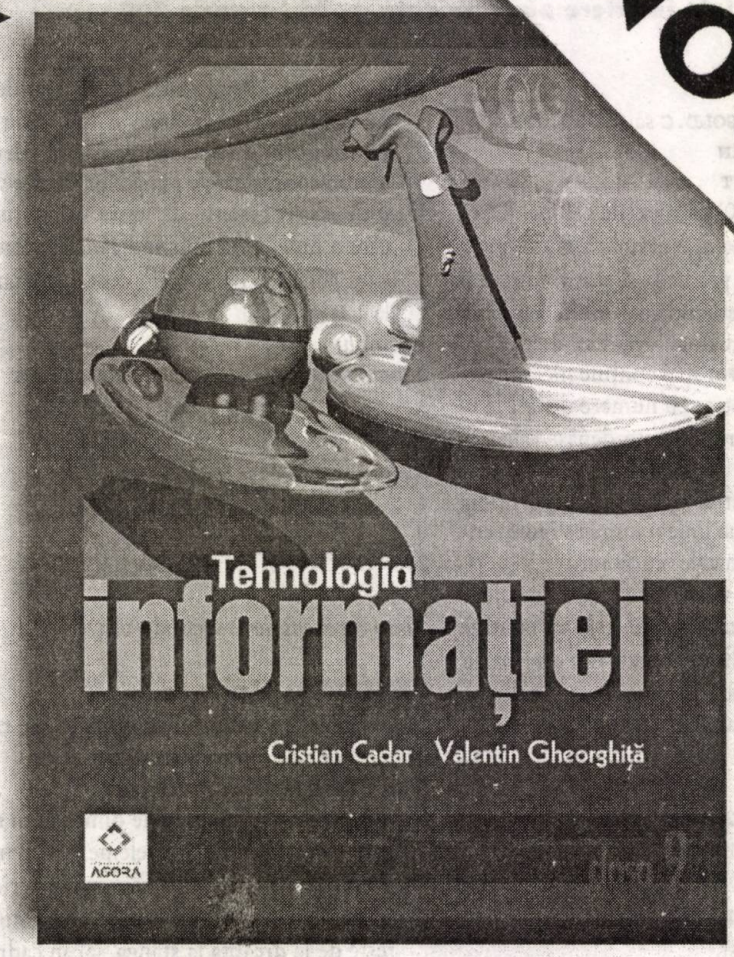
Exemplu:

`UNIT.IN`	`UNIT.OUT`
2	3
1 0	
0 1	

Timp de execuție: 3 secunde/test.

COMPUTER LIBRIS AGORA

NOUTATE



• • Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Reprezentarea BINARĂ a NUMERELOR mari - CUM și DE CE

Vlad Bazon

Lucrul cu numerele mari a reprezentat una dintre cele mai importante probleme studiate de informaticieni. În acest articol este prezentată o metodă simplă și eficientă care poate fi folosită pentru a efectua operații cu astfel de numere. Metoda prezentată se bazează pe schema lui Horner de determinare a valorii unui polinom într-un punct dat, numerele mari fiind tratate în mod asemănător unor polinoame.

Constructorul de valori numerice naturale (în baza zece) este *expresia* polinomială (*formală*)

$$a_0 + a_1*10 + a_2*10^2 + \ldots + a_n*10^n$$

unde "monoamele" a_i*10^i, $i = 0..n$, reprezintă cifrele numărului (valori cuprinse între 0 și 9) și rangurile corespunzătoare ($i = 0$ corespunde cifrei unităților).

Dacă am evalua această expresie *operând într-o altă bază*, atunci rezultatul va corespunde construcției numărului în noua bază.

Evaluarea poate decurge *analog* schemei lui *Horner* (vezi, de exemplu, [1]) - după principiul ***distribuie*** (sau "scoate în factor") ***și evaluează progresiv*** (după ordinea de imbricare dată de paranteze):

$$a_0 + 10(a_1 + 10(a_2 + \ldots + 10(a_{n-1} + 10a_n)))$$

dar calculele trebuie raportate la noua bază.

Exemplu

Pentru a determina reprezentarea în baza 50 - cu cifrele [0],[1],...,[9],[10],[11],...,[49] - a numărului 6394 dat în baza 10, vom evalua expresia

$$((6*10+3)*10+9)*10+4$$

exprimând rezultatele parțiale în noua bază:

- $6*10+3=1*50+13=[1][13]_{(50)}$;
- $[1][13]_{(50)}*10+9=13*10+10*50+9=30+2*50+10*50+9=39+12*50=[12][39]_{(50)}$;
- $[12][39]_{(50)}*10+4=39*10+12*10*50+4=40+7*50+(20+2*50)*50+4=44+27*50+2*50^2=[2][27][44]_{(50)}$

Am obținut $6394_{(10)}=[2][27][44]_{(50)}$.

Aplicație

Presupunem că tabloul CIF[] *conține cifrele zecimale (în ordinea obișnuită) ale unui număr natural; vrem să obținem reprezentarea binară a numărului respectiv și să o afișăm sub formă hexazecimală.*

De exemplu, dacă CIF[] conține cifrele numărului 123456789123456789123456789 (cu 27 de cifre zecimale), atunci rezultatul cerut este secvența de cifre hexazecimale [66][1E][FD][F2][E3][B1][9F][7C][04][5F][15], unde parantezele încadrează *câte un octet* al formei binare (evident, forma binară propriu-zisă va rezulta apoi, înlocuind fiecare cifră hexazecimală cu cei patru biți corespunzători).

Analiza problemei (CUM)

Pentru a ajunge la forma binară, ar fi suficient să determinăm octeții consecutivi ai reprezentării numărului în baza $2^8=256$ (fiecare octet furnizând, pe rând, câte 8 biți ai reprezentării binare cerute). Tabelul 1 ilustrează modul în care ar decurge evaluarea (operând în noua bază, 256), pentru expresia polinomială (în baza 10) asociată numărului 123456. Desigur, am folosit scrierea hexazecimală pentru a reda valorile octeților din tabloul CIF[]. De asemenea, am folosit convenția (specifică microprocesoarelor *Intel*) de a înregistra în ordine *inversă*, octeții componenți ai unei valori multioctet (valoarea $1E240_{16}$ se înregistrează prin secvența [40][E2][01]) - cu *consecința* clară, că reprezentarea în baza 256 care se va obține în

Pas	Calcule curente	Tabloul CIF[]
0	se dă numărul $1234..._{(10)}$	[1][2]...
1	$1*10+2=12=0C_{16}$ (1 octet)	[0C][2][3]...
2	$C_{16}*10+3=7B_{16}$	[7B][2][3][4]...
3	$7B_{16}*10+4=4D2_{16}$	[D2][04][3][4][5]...
4	$4D2_{16}*10+5=3039_{16}$ (2 octeți)	[39][30][3][4][5][6]...
5	$3039_{16}*10+6=1E240_{16}$	[40][E1][01][4][5][6][7]...
etc.		

Tabelul 1

Obținerea formei binare porționate în octeți (pentru numărul 1234...)

finalul evaluării este inversată (deci, afișarea cerută va trebui să decurgă octet cu octet, pornind *de la sfârșitul* secvenței obținute în `CIF[]`, spre început).

În tabelul 1, calculele au fost redate ca *valoare_curentă*`*10+`*următoarea_cifră*; este util - pentru a reuși o implementare simplă a algoritmului - să detaliem calculele după formula *octet_curent*`*10+`*transport*. De exemplu, la pasul 4 avem succesiv:

- *transport* ← *următoarea cifră zecimală* = 5;
- *octet_curent* ← `[D2]` (valoarea din `CIF[1]`);
- *octet_curent*`*10+`*transport* = `[D2]*10+5=839`$_{16}$ (2 octeți, `[08][39]`$_{(16)}$);
- înscrie `[39]` (adică `CIF[1]` ← $39_{(16)}$) și salvează *transport* ← 8;
- *octet_curent* ← `[04]` (valoarea din `CIF[2]`);
- `[04]*10+8=30`$_{16}$ (1 octet);
- înscrie `[30]` (nu avem transport).

În situația în care, la final, am avea *transport* ≠ 0 (cum este cazul la pasul 5), acesta trebuie înscris în locul cifrei zecimale curente (care a fost deja luată în calcul).

Această descriere amănunțită a calculului conduce la considerarea unei variabile `LoHi` de tip `Word` în care să obținem rezultatul curent, urmând să o descompunem în cei doi octeți componenți (cel *HIgh* și cel *LOw*) și să iterăm (pentru toți octeții reprezentării parțiale obținute) secvența următoare:

- `HI(LoHi)` ← cifra zecimală curentă (iar pe parcurs transportul curent);
- `LoHi` ← (*octet_curent*)`*10+HI(LoHi)`;
- `CIF[`*poziția_octetului_curent*`]` ← `LO(LoHi)`.

Astfel, revenind asupra pasului 4 din tabelul 1, am avea succesiv:

- `HI(LoHi)` ← 5, adică `LoHi = [0][5]`;
- `LoHi` ← `[D2]*10+HI(LoHi)`, adică `LoHi = [39][8]`;
- `CIF[1]` ← `LO(LoHi)=[39]`;
- avans la octetul `CIF[2]`:
  `LoHi` ← `(CIF[2])*10+ HI(LoHi)=4*10+8`, adică `LoHi=[30][0]`
  `CIF[2]` ← `LO(LoHi)=[30]`;
- încheie iterația curentă: s-a obținut reprezentarea parțială `CIF[]=[39][30]...` (urmând să se reia secvența de mai sus pentru următoarea cifră zecimală, care este acum 6).

Observație

În [2] avem o metodă de rezolvare, care este cu siguranță *mai complicată* decât aceea prezentată anterior; anume, se folosea schema lui *Horner* pentru a determina efectiv câtul și restul împărțirii prin noua bază, repetând pentru noul cât (*amestecând* schema lui *Horner* și algoritmul clasic de conversie).

Implementare

Funcțiile *Pascal* `HI(w:Word)` și `LO(w:Word)` furnizează octetul cel mai semnificativ și respectiv pe cel mai puțin semnificativ din `w`. Funcția `Swap(w:Word)` inversează octeții componenți ai valorii `w`.

Pentru scrierea în hexazecimal, am preferat să folosim funcția `HexStr(`**`var`** `num,byteCount:Byte):`**`string`**, existentă în pachetul *Borland Pascal 7.0* în directorul **EXAMPLES** (altfel, se poate căuta, de exemplu, în [1]); funcția `HexStr` returnează șirul cifrelor hexazecimale corespunzătoare celor `byteCount` octeți de la adresa `num`.

Listing APP.PAS

```
const n = 1000;      { Limitare a numarului }
                     {    cifrelor zecimale }
var CIF:array[1..n] of Byte;
    nZ,nO:Word;       {   numarul de cifre }
                      { (zecimale | octeti) }
    i,j,LoHi:Word;  { variabile de lucru - }
      { in mod normal ar trebui declarate }
      {               ca variabile locale }

function GetNr:Word;
{ incarca cifrele numarului in CIF[],     }
{ returnând numărul acestora; se poate    }
{ avea în vedere citirea dintr-un fișier  }
{ (în particular, de la CON), sau         }
{ generarea cu diverse formule (aici se   }
{ generează numărul 123...912...912...9)  }
begin
  for j:=0 to 2 do
    for i:=1 to 9 do CIF[9*j+i]:=i;
  GetNr:=27  { returneaza numarul de cifre }
             {          zecimale din CIF[] }
end;
```

```
function HexStr(var num,byteCount:Byte):
                              string;far;external;
{$L c:\bp\examples\hex.obj}

Begin
  nZ:=GetNr;
  nO:=0;                { numarul de octeti ai }
                        { reprezentarii binare }
                        {      partiale curente }
  for i:=1 to nZ do
  begin     { procedura MUL_10(...), pentru }
            {          fiecare cifră zecimala }
    LoHi:=CIF[i];
    LoHi:=Swap(LoHi);
    for j:=1 to nO do { iteratia curenta }
    begin
      LoHi := 10*CIF[j] + HI(LoHi);
      CIF[j] := LO(LoHi)
    end;
    if HI(LoHi)>0 then
     { avem transport la finalul iteratiei }
    begin
      Inc(nO);
      CIF[nO] := HI(LoHi)
    end
  end;
  for i:=nO downto 1 do
    Write('[',HexStr(CIF[i],1),']')
End.
```

Dacă $nO \leq 255$ - adică $nZ < 613$ (vezi mai jos) - atunci secvența cifrelor hexazecimale rezultată se poate afișa și direct (dar fără separatorii `'['`, `']'`) printr-un apel `Writeln(HexStr(CIF,nO))` - dat fiind că `HexStr()` ține și ea seama de faptul că reprezentarea este inversată (pornind de la octetul final `CIF[nO]`, spre primul).

Semnificația problemei (DE CE)

Este firească - față de orice program de aplicație (vezi [1]) - întrebarea: ***de ce*** "vrem să obținem...". Dacă rezultatul sau scopul aplicației nu prezintă interes pentru client (ea mizând numai pe *cum* se obține, nu și de ce), atunci am avea de-a face, mai degrabă, cu un "exercițiu aplicativ" (de învățare; cu valențe didactice și educative), decât cu o aplicație propriu-zisă.

Există două aspecte care dau sens problemei de care ne-am ocupat mai sus.

Mai întâi este clar că orice aplicație care vrea să deservească domeniul aritmeticii ("clientul" nostru, aici), apelează la numeroase iterări, imbricări și succesiuni de *operații elementare cu numere naturale*; aceste operații se desfășoară "cifră cu cifră", deci creșterea vitezei de operare poate fi asigurată și prin trecerea prealabilă (la debutul aplicației) la un sistem de reprezentare în care să fie necesare mai puține cifre (și în care să se poată lucra ulterior).

Calculatoarele operează în mod nativ cu octeți (deci în baza $2^8 = 256$); cel mai mare număr natural care are nZ cifre în baza 10 (baza cu care este obișnuit utilizatorul) este $10^{nZ}-1$, iar cel mai mare număr natural care are nO cifre în baza 2^8 este $256^{nO}-1$; ca să vorbim de aceleași numere și într-o bază și în cealaltă, trebuie să avem $10^{nZ} = 256^{nO}$, de unde

$$nZ=nO*LG(256)\approx nO*2.40824$$

Rezultă că reprezentarea binară (în baza 2^8, dacă o porționăm în octeți) va necesita de cel puțin 2.4 ori mai puține cifre (octeți) decât cea în baza 10; prin urmare, operând în baza 256 vor fi necesare de 2.4 ori mai puține operații "cifră cu cifră", decât dacă s-ar lucra direct în baza 10.

Trebuie să fie chiar "umilitor" pentru un calculator, să fie forțat să lucreze cu cifre cuprinse între 0 și 9 când el este proiectat să opereze pe o gamă de valori mult mai largă (cifre cuprinse între 0 și 255); în plus, ar mai fi necesară și modelarea în mod artificial a operațiilor în baza 10, deoarece un calculator "știe" să gestioneze transporturile față de 2^8 (și față de 2^{16}), dar nu și față de 10.

Am putea deduce, în fond, că o aplicație care folosește numere naturale mari ar trebui proiectată după schema următoare:

(i.) conversie la forma binară (porționată în octeți) a eventualilor operanzi care sunt dați în baza 10;

(ii.) algoritmul specific aplicației (pentru obținerea rezultatului), derulat folosind numai operații în baza 28 (sau multiplu de 28) - aceste operații trebuind să fie deja modelate, într-o bibliotecă preexistentă (a vedea [2]);

(iii.) eventuală reconversie a rezultatului (înapoi la baza 10).

Până acum am tratat pasul inițial (i.); propunem cititorului să trateze analog și pasul (iii.), adică să realizeze un program cât mai simplu care să scrie într-un fișier reprezentarea zecimală corespunzătoare unei reprezentări binare date (octet cu octet) în tabloul `CIF[]`.

Observație

În [3] și [4] întâlnim o altă schemă de tratare a aplicațiilor cu numere mari, folosind baza 10^4 (porțiunile de câte 4 cifre zecimale consecutive formează o "cifră") și apoi baza 2 (fiecare cifră 0000 .. 9999 este înregistrată ca un întreg pe doi bytes). În acest caz va fi necesară (în plus) prevederea gestionării transportului față de baza 10000; schema de tratare propusă mai sus - care și ea, mixează două baze: 2^8 (cu extindere aproape imediată la baza 2^{16}, sau chiar la baza 2^{32}) și baza 2 - este, în orice caz, mai naturală.

Al doilea aspect se referă la posibilitățile oferite pentru creația matematică elementară (promovată de exemplu, de către binecunoscuta revistă *Gazeta matematică*): programul poate fi utilizat pentru investigarea unor proprietăți interesante ale numerelor. Putem da un exemplu de studiu: cum se păstrează *proprietatea de egalitate a cifrelor*, trecând la altă bază de numerație?

Folosind (ca instrument de lucru) programul de mai sus, se pot sesiza (ulterior se pot eventual și demonstra și - de ce nu - se pot formula către revista de matematică, cel puțin ca "probleme propuse") proprietăți cum ar fi următoarele:

- Ultimii [n / 8] octeți ai reprezentării binare a numărului de n cifre zecimale $11\ldots1_{(10)}$ coincid respectiv cu ultimii [n / 8] termeni ai șirului periodic 28, 113, 199, 28, 113, 199, ..., 199, 28, 113, 199. (egalitatea inițială a cifrelor devine periodicitate, în noua bază). Demonstrația acestei proprietăți este bazată pe metoda inducției matematice (vezi [2]). Rezultatul s-ar exprima mai elegant în baza 8, deoarece $|28|113|199| = |1C_{(16)}|71_{(16)}|C7_{(16)}| = |07070707|_{(8)}$.
- Ultimii 8 * [n / 8] biți ai reprezentării binare a numărului de n cifre zecimale $99\ldots9$ sunt egali (toți) cu 1; reprezentarea binară a numărului 10^n se termină cu exact n biți având valoarea 0.

Bibliografie

[1] D. Rancea, G. Bălan, I. Cucu, D. Gruiță, *Informatică, manual pentru clasa a IX-a*, Editura Computer Libris Agora, 1999

[2] V. Bazon, *Limbaje și calculator*, Editura Petrion, 1998

[3] Gh. Muscă, Fl. Moraru, Fl. Munteanu, *Informatică aplicată (manual pentru liceele de informatică, cl. a XI-a, alternativa A)*, Editura Didactică și Pedagogică, 1998

[4] H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Birkhuser, 1985

Dl. prof. Vlad Bazon este cadru didactic la Liceul "Anghel Rugină" din Vaslui. Poate fi contactat prin intermediul redacției.

news..

Netscape 6.0

Mult așteptata versiune 6.0 a popularului program de navigare produs de Netscape este, în sfârșit, disponibilă.

În mod normal, numărul acestei versiuni ar fi trebuit să fie 5.0, dar, deoarece programul de navigare Internet Explorer se află deja la versiunea 5.5, Netscape a dorit să fie în fața companiei rivale cel puțin din acest punct de vedere; așadar, compania a ales să sară peste versiunea 5. Această practică a fost folosită și de alte companii printre care și rivalii de la Microsoft.

Noul browser a fost rescris aproape în întregime pentru a avea o dimensiune cât mai mică oferind, în același timp, o mulțime de facilități.

A rezultat un produs puternic care poate fi descărcat într-un timp rezonabil chiar și prin intermediul unui modem obișnuit.

A fost creat programul Netscape 6 Installer care permite descărcarea componentelor specifice care se doresc instalate, fără a se mai pierde timpul cu cele care nu sunt folosite.

A fost introdus și un buton de căutare, care are un rol similar cu cel din Internet Explorer.

Noul Netscape 6 oferă și facilitățile unui program care gestionează mesajele de poștă electronică. Există posibilitatea gestionării mai multor conturi e-mail.

Programul de navigare oferă diverse posibilități de personalizare, interfața putând fi foarte ușor adaptată nevoilor fiecărui utilizator.

Deoarece această versiune a întârziat mai mult de un an, Netscape a încercat să mențină interesul pentru programele de navigare proprii lansând versiuni succesive bazate pe Netscape Communicator 4.5. Au existat o mulțime de variante ale acestui produs, cele mai importante fiind 4.51, 4.6, 4.61, 4.63, 4.7, 4.72, 4.74, 4.75 și 4.76.

Chiar dacă Netscape susține că noul program de navigare va marca începutul unei noi revoluții în Internet, această versiune nu oferă decât facilități de care utilizatorii Internet Explorer beneficiau încă de acum un an.

Netscape 6 a fost primul produs important open-source al companiei. La realizarea formei finale au contribuit programatori din întreaga lume.

Mulți afirmă că Netscape 6 a fost lansat în grabă și ultimele bug-uri semnalate nu au fost corectate. Din aceste motive nu se recomandă folosirea versiunii curente; poate că ar fi mai bine să așteptăm o versiune care nu va conține erori.

Produsul Netscape 6 poate fi descărcat de la adresa www.netscape.com.

INTEL revine

Mihai Scorțaru

Pe data de 20 Noiembrie 2000 compania Intel a lansat o nouă familie de procesoare bazată pe o arhitectură nouă și, după părerea multora, a redevenit lider în războiul procesoarelor. În acest articol veți găsi câteva detalii referitoare la noul procesor, precum și informații despre procesoarele mai vechi.

Intel Pentium 4

După mai multe amânări, procesorul *Intel Pentium 4* poate fi acum achiziționat de oricine își permite să cheltuie aproape 1000 de dolari pentru a avea ultima noutate apărută pe piață.

Acest procesor se bazează pe o arhitectură nouă numită *NetBurst*. *Intel* nu a mai modificat arhitectura de bază a procesoarelor sale de la lansarea familiei *Pentium Pro*. Vechea arhitectură se afla la capătul liniei de evoluție, astfel încât folosind actuala tehnologie, este aproape imposibilă atingerea de viteze care depășesc 1.2 GHz. Trecerea la tehnologia de 0.13 microni ar mai putea aduce doar câteva sute de megahertzi în plus. Încercări de a construi procesoare cu viteze mai mari nu ar duce decât la creșterea cantității de căldură degajată de procesor, și nu la creșteri reale de performanță. Comparând actualul *Pentium III* la 1 GHz cu vechiul *Pentium Pro*, observăm o creștere a performanțelor de aproape zece ori. Noua arhitectură este la începutul liniei de evoluție și ne putem aștepta ca *Intel* să fie capabilă să producă procesoare de până la 10 GHz, fără să mai schimbe arhitectura. Actualul procesor *Pentium 4* conține aproximativ 42.000.000 de tranzistori, motiv pentru care dimensiunile sale (deci și costurile de producție) s-au mărit.

Intel Pentium 4 este disponibil la frecvențe de 1.4 și 1.5 GHz și are o magistrală cu frecvența de 400 MHz. De asemenea, a apărut un nou set de instrucțiuni denumite SSE2. Cele 144 de instrucțiuni ale acestui set extind tehnologiile MMX și SSE.

Procesorul are un motor de execuție rapid, astfel încât unitatea aritmetică și logică lucrează la o viteză de două ori mai mare decât miezul procesorului, deci ALU va funcționa la 2.8 respectiv 3 GHz pentru procesoarele disponibile în acest moment.

Pentium 4 a fost calificat ca fiind cel mai controversat procesor *x86* din toate timpurile. După lansarea acestuia opiniile analiștilor au fost împărțite. Comportamentul neașteptat al procesorului în diferite *benchmark*-uri aduce confuzie în rândul specialiștilor. Majoritatea articolelor apărute în presa americană după această lansare au tras concluzii negative, deoarece comentatorii nu puteau vedea noul procesor ca fiind atât de extraordinar cum ar vrea *Intel* să îl prezinte. Mulți se așteptau ca lansarea acestui procesor să fie un eșec așa cum a fost cea a procesorului *Pentium III* la 1.13 GHz. Se pare că *Intel* a învățat lecția și nu a mai lansat pe piață un produs necompetitiv.

Chiar dacă *Pentium 4* este procesorul cu cea mai ridicată frecvență de pe piață, se pare că nu este cel mai performant. În multe *benchmark*-uri un *Athlon Thunderbird* la 1.2 GHz este superior unui *Pentium 4*, iar, în unele, chiar și un *Pentium III* la 933 MHz depășește un *Pentium 4*.

Planurile Intel

S-ar putea ca în primul trimestru al anului următor să apară și un *Pentium 4* la 1.3 GHz pentru a umple golul între *Pentium III* la 1 GHz și *Pentium 4* 1.4 GHz.

Tot la începutul anului 2001 procesoarele *Celeron* vor avea magistrala de 100 MHz. Primul procesor de acest tip va avea frecvența de 800 MHz. Varianta *Celeron 850* va apărea în al doilea trimestru al anului 2001.

În al doilea trimestru sunt așteptate procesoare *Pentium 4* cu frecvențe de 1.7 și 2 GHz. Unii sunt convinși că aceste procesoare vor apărea în primul trimestru, existând zvonuri potrivit cărora vom avea procesoare *Pentium 4* la 2 GHz în ianuarie 2001.

Tot în al doilea trimestru al anului viitor se preconizează lansarea procesorului *Pentium 4 Xeon*. Acesta poartă acum numele de cod *Foster* și va fi lansat la o frecvență de 1.7 GHz.

În același trimestru se preconizează reapariția procesorului *Pentium III* la 1.13 GHz, dar mulți sunt convinși că această relansare nu își are rostul.

De asemenea, se pare că *Intel* va trece la tehnologia de 0.13 microni și va lansa procesoare *Pentium III* bazate pe

aceasta. Acestea poartă numele de cod *Tualatin* și vor apărea la frecvențe de cel puțin 1.26 GHz.

Succesorul actualului *Pentium 4* poartă numele de cod *Northwood*. Acesta va beneficia și el de avantajele tehnologiei de 0.13 microni și va fi lansat spre sfârșitul anului următor.

Procesoarele *Mobile Celeron*, destinate calculatoarelor portabile, vor atinge 750 MHz în primul trimestru și 800 MHz în cel de-al doilea.

Notebook-urile care folosesc *Mobile Pentium III* vor beneficia de procesoare la frecvențe de 900 MHz în primul trimestru și 1 GHz în al doilea.

Varianta procesorului *Tualatin*, destinată calculatoarelor portabile, va apărea în același timp cu cea destinată *desktop*-urilor.

Planurile AMD

Numele de cod al următorului procesor *Athlon* de la *AMD* este *Palomino*. Probabil că aceste procesoare vor apărea la începutul anului 2001 și, inițial, vor avea frecvența de 1.33 GHz. În al doilea trimestru va apărea varianta la 1.5 GHz, iar în a doua jumătate a anului cea la 1.7 GHz.

Palomino va apărea și într-o variantă destinată calculatoarelor portabile. Aceste procesoare vor beneficia de tehnologia *PowerNow!*.

Următoarea generație de procesoare *Duron* va apărea în primul trimestru al anului 2001 la o frecvență de 850 MHz. Aceste noi procesoare poartă numele de cod *Morgan*. Al doilea trimestru ne va aduce procesoare *Morgan* la 900 MHz, iar bariera de 1 GHz va fi atinsă în a doua jumătate a anului.

Va apărea și o variantă pentru notebook-uri care va beneficia și ea de tehnologia *PowerNow!*.

La începutul anului 2002 *AMD* va trece la producția de procesoare bazate pe tehnologia de 0.13 microni. Versiunea procesorului *Athlon* bazată pe această tehnologie poartă numele de cod *Thoroughbred* și va atinge frecvența de 2 GHz doar în a doua jumătate a anului 2002. Vor fi și versiuni pentru calculatoare mobile.

Vor fi produse și procesoare *Duron* bazate pe tehnologia de 0.13 microni; acestea poartă numele de cod *Appaloosa* și vor apărea atât în variante pentru desktop-uri cât și pentru notebook-uri.

Primul procesor care va folosi arhitectura *AMD* din a opta generație, poartă numele de cod *ClawHammer*; acest procesor folosește o arhitectură pe 64 de biți. *ClawHammer* va fi disponibil la începutul anului 2002. Varianta pentru servere a acestui procesor poartă numele de cod *SledgeHammer* și va apărea în al doilea trimestru al anului 2002.

Calculatoare performante

Pe piața procesoarelor destinate calculatoarelor performante singura noutate o reprezintă procesorul *Intel Pentium 4*. În rest prețurile au scăzut puțin față de luna trecută păstrând o tendință normală. Se așteaptă reduceri de prețuri pentru procesoarele *Pentium 4* în cursul lunii decembrie. Următoarele reduceri de prețuri ar putea avea loc la începutul anului următor.

Figura 1 prezintă graficul prețurilor procesoarelor *Pentium III* cu magistrala de 133 MHz și ale procesoarelor *Athlon Thunderbird*. Se observă că prețurile practicate de compania *AMD* sunt mai mici.

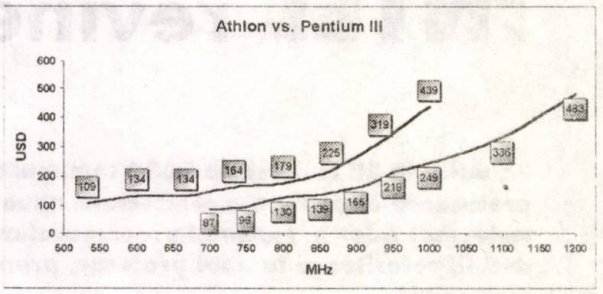


Figura 1

Sunt disponibile pe piață și procesoare *Pentium 4*, dar prețurile acestora sunt foarte mari. Un *Pentium 4* la 1.4 GHz costă 760$, în timp ce pentru un *Pentium 4* la 1.5 GHz trebuie plătiți nu mai puțin decât 915$.

Calculatoare ieftine

Pe piața procesoarelor destinate calculatoarelor ieftine au apărut două noi produse ale firmei *Intel*, și anume procesoarele *Celeron* cu frecvențele de 733 și 766 MHz. Totuși, în acest moment, procesoarele *Celeron* de la *Intel* nu se pot compara cu procesoarele *Duron* de la *AMD*, chiar și un *Duron 600* fiind, în majoritatea cazurilor, mai rapid decât un *Celeron 766*.

Prețurile acestor procesoare sunt relativ mici, doar noile procesoare *Celeron* depășesc suma de 100$, după cum se poate vedea și în graficul din figura 2. Din nou, produsele *AMD* sunt mai ieftine.

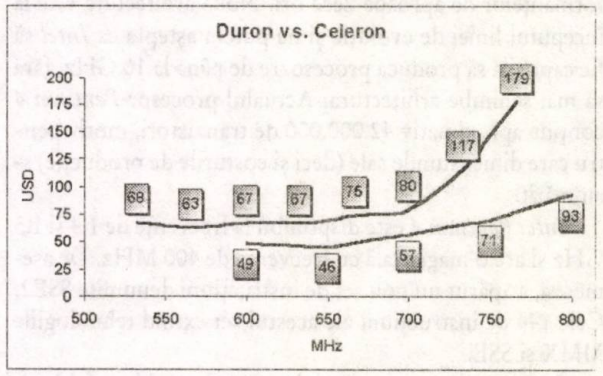


Figura 2

Concluzie

Anul acesta piața procesoarelor a avut o dinamică fără precedent. Pentru prima dată a apărut o companie care a reprezentat un adevărat pericol pentru *Intel*. Cu siguranță *AMD* nu și-a spus încă ultimul cuvânt în războiul procesoarelor.

Aplicații MULTIMEDIA în procesul EDUCAȚIONAL

Manuela Mihăescu

A ști să înveți, a ști să te informezi este un nou tip de comportament într-o societate nouă, în care principalele "materii prime" sunt informația, comunicarea. Noile tehnologii de comunicare și informare constituie, în ansamblul lor, una dintre sfidările actuale ale societății noastre. Dacă unii dintre termenii ca CD, Internet, hypertext, interactiv etc. sunt destul de cunoscuți, fenomenul în sine și obiectivul acestui fenomen rămân încă destul de obscure pentru majoritatea celor neinițiați.

Noile tehnologii de stocare, prelucrare și comunicare au dat naștere unei noi modalități de tratare a informației care permit căutarea, citirea, copierea, colarea, organizarea și trimiterea informației în moduri variate, sigure și rapide. În educație, datorită acestor tehnologii apar forme noi de predare și de învățare. Având în vedere posibilitatea comprimării datelor (fiind din ce în ce mai ușoară utilizarea și stocarea lor), combinate cu digitalizarea și transmiterea prin fibre optice, aceste tehnologii, dacă sunt înțelese și aplicate corect, pot avea un mare potențial în formare, propunând un viitor foarte promițător cu reale capacități de transformare a procesului educațional.

Dicționare, enciclopedii și e-cărți

Cărțile au reprezentat din cele mai vechi timpuri instrumentele principale de învățare. Odată cu progresul tehnologic, au apărut alte modalități de informare, cum ar fi filmele documentare sau materialele audio, care permiteau învățarea mult mai rapidă, datorată diversificării formelor de prezentare a materialelor: combinații de imagini, filme, sunet, muzică. În ultimii ani, progresul tehnologic informatic a permis apariția enciclopediilor, dicționarelor și cărților *electronice*. Acestea realizează un salt uriaș în comparație cu vechiul, dar încă actualul și omniprezentul format - cartea simplă, clasică. Câteva din avantajele majore ale formatului electronic sunt:

- materialele multimedia, ca mijloc de prezentare (filmele exemplificative, animații pentru explicarea unor procese, sunete pentru pronunții sau fenomene, muzică etc.);
- căutarea rapidă a unui fragment sau cuvânt dorit;
- *hiperlink*-urile - legături electronice care conduc rapid la anumite explicații, referiri, capitole;
- posibilitatea de a copia (captura) un fragment dorit într-un editor de text sau de imagine, în scopul unei documentări, cercetări, fiind oricând posibilă tipărirea materialului;
- posibilitatea de a adăuga informații materialelor electronice (de exemplu, introducerea unor cuvinte sau expresii noi într-un dicționar);
- accesul la ultimele noutăți (aducerea "la zi" a enciclopediei), încărcând *add-on*-uri de pe *Internet*.

Cartea electronică a apărut deja în librării, figurează în cataloagele editorilor renumiți, poate fi expusă în standurile din diversele saloane de cărți. Se poate procura pe o dischetă, CD sau în direct prin intermediul *Internet*-ului. Se poate consulta direct pe ecran sau se poate imprima pe hârtie. Dezvoltarea rapidă se explică prin facilitatea oferită în raport cu cărțile tipărite și bineînțeles datorită progresului tehnologic care oferă posibilitatea consultării foarte rapide.

Cărțile electronice sunt scrise și se consultă folosind calculatorul. Ele se citesc, de obicei, cu ajutorul unui editor de texte sau editor *HTML*, pentru lecturare fiind necesar rereori un program special. Conținutul acestora este în cea mai mare parte text, cu posibile inserții de imagine sau, mai rar, de sunete sau filme. În marea majoritate apar *hiperlink*-urile care ușurează foarte mult lectura. În ziua de astăzi, pe *Internet* apar din ce în ce mai multe biblioteci virtuale, de unde se pot achiziționa, uneori contra cost, alteori gratuit, cărți sau reviste electronice. Iată câteva adrese de acest fel din țara noastră:

- Editura *Nemira*: www.nemira.ro (permite "încărcarea" gratuită a trei cărți electronice de literatură SF);

- Editura *RAO*: www.rao.ro;
- Magazinul virtual *Teora*: www.teora.ro;
- Librăria on-line *Cartea*: www.cartea.ro.

Multe dintre aceste magazine virtuale permit cumpărarea prin *Internet* a numeroase cărți clasice, trimise prin poștă, cu plata ramburs. Recent, au început să apară cărți doar în format electronic, (de exemplu, ultima carte a cunoscutului scriitor american *Stephen King* intitulată ***Riding the Bullet***, se poate achiziționa numai în format electronic de pe *Internet*, neexistând și varianta clasică tipărită).

O alternativă (mai accesibilă) la cărțile de specialitate, format clasic (sau cursurile de calculatoare, relativ scumpe) ar fi site-urile de tip "librărie virtuală" în care sunt oferite gratuit documentații sau cursuri întregi din domenii variate: grafică, DTP, sisteme de operare, limbaje de programare, întreținerea PC-ului (***Photoshop în 14 zile***, ***Linux***, ***Visual Basic***, ***Delphi*** etc.). Acestea se consultă de obicei în direct pe site-ul respectiv.

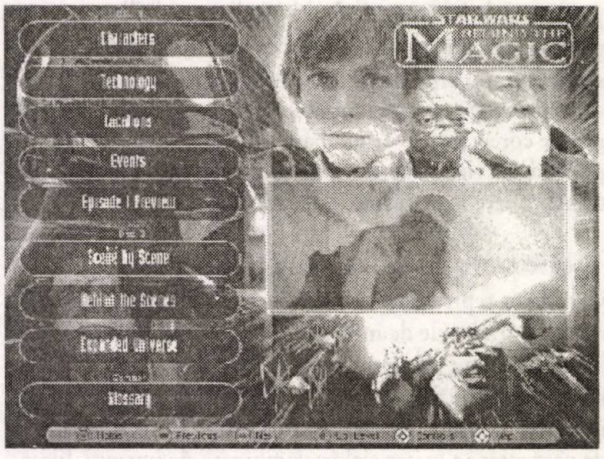
Dicționarele electronice au un avantaj incontestabil față de cele clasice deoarece căutarea unui cuvânt sau a unei expresii are loc practic instantaneu, există posibilitatea de pronunțare/audiere a materialelor, corectarea sau traducerea automată a unui fragment de text și îmbogățirea prin introducerea unor cuvinte sau expresii noi a dicționarului. Ca posibilități de corectare automată se întâlnesc: corectare din punct de vedere sintactic, corectare gramaticală, oferindu-se diferite sugestii pentru cuvintele necunoscute, crearea unor dicționare proprii și toate acestea pentru diferite limbi.

Enciclopediile electronice reprezintă cel mai complex și mai fascinant material educațional. De obicei, acestea sunt stocate pe unul sau mai multe CD-uri, în funcție de cantitatea informațiilor multimedia (filme, materiale audio, muzică), textul ocupând cel mai puțin spațiu. Există la ora actuală enciclopedii în toate domeniile posibile. Câteva dintre cele mai cunoscute și utilizate enciclopedii:

- Enciclopedia ***Britannica*** - 3 CD-uri care înglobează toate volumele celebrei enciclopedii engleze, fiind cea mai cunoscută și completă sursă de informație. Varianta online pune la dispoziție toate resursele conținute în volumele tipărite, precum și numeroase articole din diverse publicații și link-uri către site-uri tematice. În momentul în care se caută un termen, sunt oferite, ordonate în funcție de relevanță, explicațiile de dicționar, textele din enciclopedie, cele mai bune site-uri care au legătura cu acel subiect, link-uri către cărți tipărite sau on-line.
- Enciclopedia ***Encarta*** - o enciclopedie universală deosebit de complexă și atractivă, realizată de cunoscuta firmă *Microsoft*;
- Enciclopedia ***Learn to speak English 8*** - un profesor de engleză pe 4 CD-uri; elevul beneficiază de un profesor virtual, care apare într-o fereastră animată a programului, poate comunica verbal cu acesta prin intermediul microfonului și va fi imediat corectat dacă greșește, poate rezolva sute de exerciții gramaticale sau consulta un impresionant dicționar;
- Enciclopedia ***Cinemania*** - toate informațiile despre filme, actori, regizori;
- Enciclopedia ***Art Gallery*** - tablourile din cel mai vestit muzeu londonez, informații despre pictori și alte informații curente din pictură;
- Enciclopedia ***Billboard Music Guide*** - informații despre albume de muzică din toate genurile, despre artiști;
- Enciclopedia ***World Atlas*** - harta lumii în cele mai mici detalii cu referiri la obiceiuri și tradiții locale etc.

Există enciclopedii axate pe anumite domenii speciale de interes, cum ar fi: chinologie, aviație etc. sau pe diferite subiecte de actualitate, cum ar fi, de exemplu, seria filmelor ***Războiul stelelor***. Toate acestea, pornind de la texte bine documentate, cu *hyperlink*-uri, oferă un buchet impresionant de filme, animații, muzică sau sunete care exemplifică fiecare amănunt. Ele conțin de asemenea indexuri structurate pe arii de interese (sau generale) și dicționare explicative ale cuvintelor întâlnite. De exemplu, în enciclopedia ***Cinemania***, tastând numele unui actor, apare biografia acestuia, toate filmele în ordine cronologică în care a jucat (unele fiind însoțite de imagini, sunet și muzică), premiile pe care le-a obținut, cu cine a colaborat, se poate "sări" la unul din filme pentru a citi recenzii asupra acestuia sau vizualiza un fragment în format video.

Un alt exemplu este Enciclopedia ***Star-Wars - Behind the Magic***, care tratează cele patru filme cunoscute, personaje, tehnologie, locații, evenimente, explicarea fiecărei scene din filme sau a modului de realizare a filmelor. În fereastra din mijloc rulează fragmente video.



Meniul principal al enciclopediei Star Wars - Behind the Magic

Este de remarcat faptul că toate aceste materiale multimedia necesită un calculator dotat cu placă video cât de cât performantă, CD-ROM, placă de sunet, microfon, boxe etc.

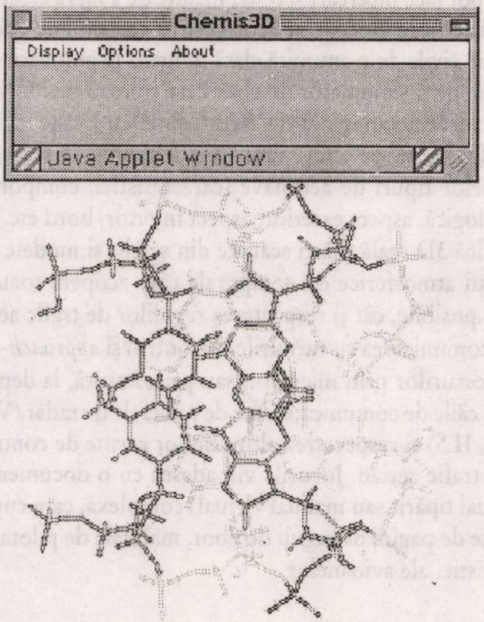
Simulări, jocuri virtuale - un mod de a învăța...

Prin simulările oferite de unele programe, în care cifrele și textul (parametrii) pot fi personalizați în funcție de dorințe, se trece la un alt mod de reprezentare a realului,

obținându-se astfel un imens univers (care depășește adesea domeniul informaticii) și care deschide o nouă eră a reprezentării lumii fizice și lumii mentale. Se pot simula diferite forme, statistici, reacții, realitatea virtuală fiind o manieră prin care poate fi reprezentată lumea reală.

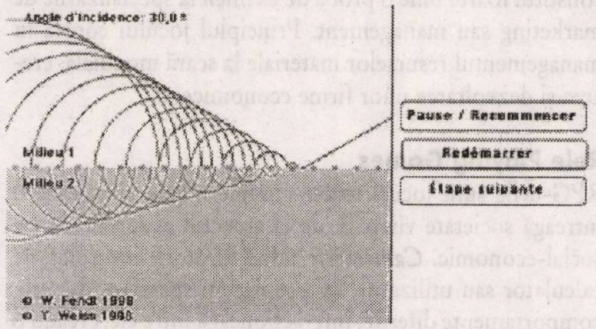
În ceea ce privește formarea și educarea, utilizarea calculatorului și-a câștigat un rol hotărâtor, mai ales în ceea ce privește învățarea activă, care permite dobândirea de "gesturi profesionale", formarea de deprinderi. Se constată astăzi că se practică din ce în ce mai mult interactivitatea on-line pe *Internet*, laboratoarele virtuale, în care utilizatorul poate interveni, modificând anumiți parametri pentru a studia diferitele reacții, fenomene etc.

De exemplu, grupul de programe denumit *Chemis3D* permite vizualizarea structurii chimice în trei dimensiuni în cadrul unei pagini web. Modelele pot fi manipulate în timp real și aduse la forme diferite.



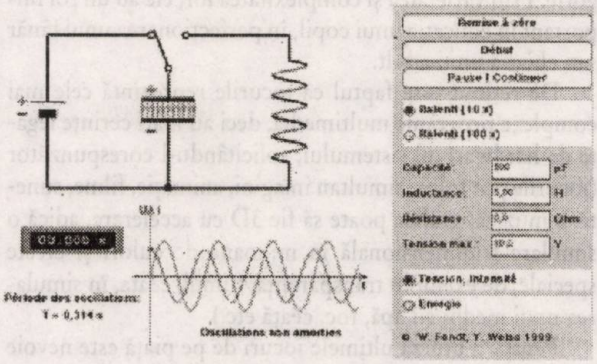
Programul Chemis3d

Programul *Fizică-refracția* explică fenomenul de reflexie și refracția undelor pe baza principiului lui *Huygens*. Există posibilitatea modificării indicelui de refracție și a unghiului de incidență.



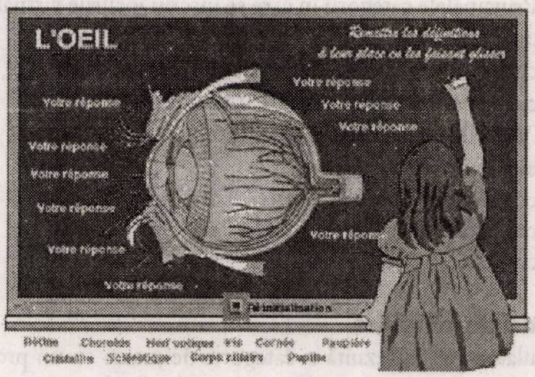
Programul Fizică-refracția

Programul *Fizică-electricitate* este un simulator al unui circuit electromagnetic oscilant constituit dintr-un condensator și o bobină. Prin activare, plăcile condensatorului se încarcă, cea de sus pozitiv, iar cea de jos negativ. Selectând "Start" începe oscilația. Se pot modifica valorile capacității condensatorului, inductanței și rezistenței bobinei, ca și tensiunea eliberată de generator.



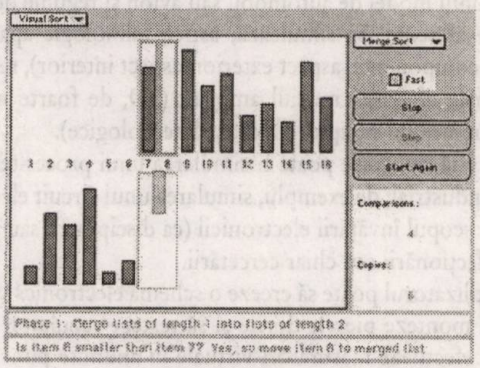
Programul Fizică-electricitate

Pentru acumularea de cunoștințe, în ceea ce privește, de exemplu ochiul, programul *Biologie* propune, deplasarea cu ajutorului mouse-ului a răspunsului corect și localizarea lui pe desen.



Programul Biologie

Aplicația *Algoritmi de sortare informatică* permite observarea algoritmilor de sortare și evaluarea eficacității lor. Fiecare etapă constituie obiectul unui comentariu iar animația este foarte sugestivă.



Programul Algoritmi de sortare informatică

Jocurile, în general, reprezintă o modalitate plăcută de ocupare a timpului liber, preferată atât de tineri cât și de mulți adulți.

După gradul de solicitare ele se împart în două categorii: jocuri care implică fizic organismul și jocuri care se adresează intelectului.

Jocurile pe calculator fac parte din această ultimă categorie. Prin varietatea și complexitatea lor, ele au un rol important în educația unui copil, în perfecționarea unui tânăr sau chiar a unui adult.

De reținut este faptul că jocurile reprezintă cele mai complexe programe multimedia, deci au mari cerințe legate de hardware-ul sistemului, solicitându-l corespunzător (jocurile pot folosi simultan imagini, animație, filme, sunete și muzică; grafica poate să fie 3D cu accelerare, adică o simulare tridimensională în milioane de culori și efecte speciale, de exemplu transparența, fumul, ceața, în simularea unui mediu cu apă, foc, ceață etc.).

Pentru a utiliza ultimele jocuri de pe piață este nevoie de un calculator dotat cu un procesor rapid, multă memorie RAM, placă grafică cu accelerare 3D, placă de sunet, CD-ROM rapid, un sistem de amplificare și redare a sunetului, și, nu în ultimul rând, un monitor capabil să afișeze rezoluțiile dorite, placă de rețea sau un modem pentru jocuri în rețea.

Principalele categorii în care se înscriu jocurile virtuale sunt:

- jocurile de simulare (simulatoarele)
- jocurile de strategie (tactică sau economică)
- RPG-urile (*role-playing-games*)
- jocurile de aventură
- jocurile de acțiune
- jocurile de sport
- jocurile de societate

Jocuri de simulare

Simulatoarele reprezintă o categorie de jocuri cu un procent educativ extrem de ridicat. Jocurile simulează practic orice aparat sau proces. Progresul tehnologic duce la creșterea continuă în timp a realismului simulării, de la aspectul grafic la cel științific.

Simulatoarele cel mai des folosite sunt cele de automobile și aeronave. Orice utilizator poate învăța rapid conducerea unui model de automobil sau avion și regulile de circulație aferente. Se simulează, astfel, tehnologic aparatul (piese componente, aspect exterior, aspect interior), regulile sociale aferente, mediul ambiant (3D, de foarte multe ori), vremea (zi-noapte, condiții meteorologice).

O altă categorie poate fi simularea unui proces tehnologic industrial, de exemplu, simularea unui circuit electronic, în scopul învățării electronicii (ca disciplină), sau acela al perfecționării sau chiar cercetării.

Utilizatorul poate să creeze o schemă electronică virtuală, să monteze piesele dorite și să testeze rezultatul final sau să efectueze măsurători virtuale în anumite puncte de control.

Simularea unui zbor în apropierea orașului Seattle cu avionul Beechjet

Astăzi există mii de simulatoare în toate domeniile posibile, uneori de un realism extrem de ridicat. Cu ajutorul lor se pot face încercări și teste, înainte de a lucra efectiv cu modelele reale. Recent, se acordă chiar carnetul de pilot pe avioane civile de o anumită clasă în urma promovării unui examen pe un simulator de zbor care rulează pe PC, accesibil oricărui cumpărător. Acest simulator este atât de complex încât, pe lângă simularea tehnologică perfectă a diverselor tipuri de aeronave (caracteristici, componență tehnologică, aspect exterior, aspect interior, bord etc.), are o grafică 3D reală (hărți scanate din satelit și modele 3D), condiții atmosferice ori temporale care acoperă toate cazurile posibile, cât și respectarea regulilor de trafic aerian, de la comunicarea cu turnurile de control și *approach*-urile aeroporturilor prin microfon sau pe tastatură, la deplasarea pe căile de comunicații date de punctele de radar (VOR, NDB, ILS) și respectarea altitudinilor cerute de controlorii de trafic aerian. Jocurile vin adesea cu o documentație (manual tipărit sau manual virtual) complexă, care cuprinde sute de pagini de reguli de zbor, manuale de pilotaj, caracteristici ale avioanelor.

Jocuri de strategie

Jocurile de strategie stimulează gândirea tactică-strategică și economică a utilizatorului. În jocul *Civilisation*, acesta trebuie să fie un bun diplomat, să angajeze discuții, să facă un comerț în profit, să construiască planuri de dezvoltare economică și socială în anumite orașe, să învețe și să aplice tactici de strategie militară. Jocul *Capitalism Plus* ar putea constitui foarte bine o probă de examen la specializările de marketing sau management. Principiul jocului constă în managementul resurselor materiale la scară mondială, crearea și dezvoltarea unor firme economice.

Role Playing Games

RPG-urile sunt jocuri foarte complexe care simulează o întreagă societate virtuală, de la aspectul geografic, la cel social-economic. *Caracterele*, adică jucătorii controlați de calculator sau utilizator, au proprietăți specifice, meserii, comportamente diferite, interacționează între ele, creând o adevărată lume virtuală.

Caracteristicile, ca de exemplu forța, îndemânarea, inteligența etc., se schimbă în mod continuu, în funcție de modul de abordare a jocului și influențează mersul evenimentelor. Jucătorul poate să poarte discuții cu personajele controlate de calculator, să se alieze sau să lupte cu alții, să inițieze schimburi, fiecare cuvânt sau gest putând avea o deosebită importanță în desfășurarea jocului, la fel ca și în viața reală. De aceea, RPG-urile solicită inteligența utilizatorilor și aceștia își pot da seama de consecințele unui anumit comportament ales, la fel ca și într-un adevărat studiu psihologic.

Jocuri de aventură

Jocurile de aventură solicită inteligența, perspicacitatea și inventivitatea utilizatorilor în rezolvarea anumitor enigme sau a unor puzzle-urilor. În jocuri se întâlnesc personaje controlate de calculator cu care trebuie purtate discuții, sau obiecte care au un rol predefinit, care se pot combina sau cu care se poate manipula sau interacționa la un moment dat. RPG-urile și jocurile de aventură seamănă foarte mult cu o construcție de roman complex. Ele prezintă, de la început, un fir epic, care se va dezvolta, se va amplifica pe parcursul jocului în funcție de modul de abordare al jucătorului, fir epic ce poate induce, prin combinare, adevărate drame, suspans-uri, lovituri de teatru, rezolvări fericite. Dialogurile din joc sunt de obicei complet "vorbite" (de către actori profesioniști), iar între misiuni apar filme animate care creează o atmosferă deosebită și dau o mare savoare jocului.

Jocuri de acțiune

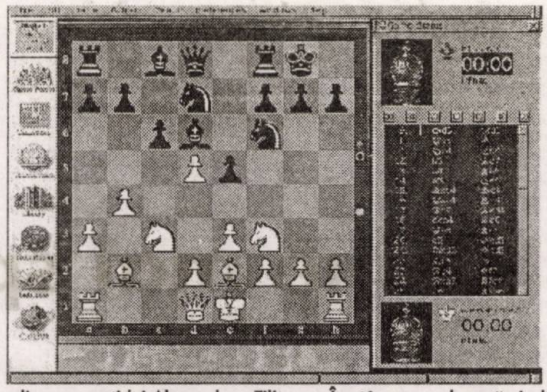
Jocurile de acțiune se bazează pe îndemânarea, spiritul de reacție și manipularea rapidă și eficientă a tastaturii, mouse-ului sau joystick-ului folosit. Cea mai întâlnită categorie, shooter-ele (trad. shoot = a împușca) sunt jocuri de luptă, de obicei într-un mediu 3D, având cerințe mari privind resursele sistemelor utilizate. Aceste jocuri se adresează în special copiilor, dezvoltându-le îndemânarea și luarea rapidă de decizii.

Jocuri de sport și de societate

Jocurile de sport și jocurile de societate sunt un fel de simulatoare a sporturilor existente și ale jocurilor de cărți, ale celor pe diverse table de joc, aparate de cazino etc. Jocurile de sport dezvoltă mai cu seamă îndemânarea, cele de societate mai mult inteligența jucătorilor.

De exemplu, jocurile de șah au atins performanțe extraordinare în competițiile cu jucători profesioniști, câștigând chiar în fața campionului mondial de șah.

Un bun program de șah are o bibliotecă dotată cu sute de mii de partide de șah cunoscute, și comentate de specialiști, o teorie a deschiderilor, a jocurilor de mijloc și în final, analize poziționale ale unei mutări, diferite grade de dificultate, de la începător la mare maestru internațional, simulează un turneu de șah cu diverse personalități (stiluri diferite de joc) ș.a.m.d.



Analizarea partidei Alexandrov-Filippov. În stânga se observă și alte opțiuni în studierea șahului.

De remarcat este faptul că, majoritatea jocurilor reprezintă adevărate programe de inteligență artificială. Complexitatea lor este adeseori uriașă. Echipe de zeci, chiar sute de oameni, lucrează ani de zile la un joc mai complex. În aceste echipe colaborează programatori, designeri, graficieni, actori, muzicieni cu regizori, scriitori, specialiști în diverse domenii. De multe ori firma (firmele) care dezvoltă jocul, colaborează cu alte instituții, fabrici, laboratoare, de la care primesc consultanță.

În jocurile pentru rețea interacționează mai mulți participanți. Există rețele locale, în care "concurenții" se pot "înfrunta", sau, utilizând *Internet*-ul, jucători din întreaga lume pot să se întâlnească în orașe virtuale.

Concluzii

Succesul jocurilor video care exploatează această dimensiune a noilor tehnologii informaționale atestă caracterul deosebit de atractiv al acestora.

Am realizat aproape o pledoarie a jocurilor. Prin fascinația pe care o exercită asupra utilizatorului, nedându-i nici un moment impresia efortului intelectual (chiar dacă acesta este prezent pe toată durata procesului ludic), jocurile - ca aplicații ale noilor tehnologii informaționale - pot fi foarte ușor convertite în mijloace de învățare rapidă și eficientă. Ce dimensiuni nebănuite și pasionante poate dobândi istoria de pildă, într-o succsiune de programe care i se adresează elevului, îl solicită ca interlocutor, explorator virtual al lumilor apuse și reînviate. Subliniem că realitatea virtualului nu trebuie să copleșească realitatea vie, iar jocului de-a/cu istoria trebuie să-i succeadă mereu reîntoarcerea în real.

Nu putem încheia fără a trage semnalul de alarmă: în timpul unui joc, mai ales individual, timpul trece pe neobservate. Este un timp care nu se va întoarce niciodată... Atenție, deci: nu exagerați, chiar dacă aveți în calculator un partener cu răbdări infinite, care nu vă reproșează greșelile și care este în stare să vă suporte reluând jocul de nenumărate ori.

D-na Manuela Mihăescu este documentarist la Facultatea de Litere din cadrul Universității Babeș-Bolyai din Cluj. Poate fi contactată prin e-mail la adresa **manuela@lett.ubbcluj.ro**.

Ce ne mai spun SURSELE?

Mihai Oltean, Crina Groșan

În cadrul acestui articol vom arăta că algoritmii nu sunt chiar așa de reci precum par. Ei reușesc, uneori, - dacă îi scriem cum trebuie - să transmită mai mult decât suma instrucțiunilor din care sunt alcătuiți. Spre exemplu sentimente, gânduri de prietenie, rețete de mâncare, meciuri de fotbal, povești pentru copii etc. Noi ne-am limitat la câteva exemple sumare, dar folosind instrucțiunile dintr-un limbaj de programare (care la prima vedere par stricte și rigide) se pot naște universuri mai fine decât o atingere de mână.

O bucată de granit aflată la o margine de drum e rece. Oamenii trec pe lângă ea și nu o bagă în seamă. Eventual, dacă sunt obosiți, se așează pe ea. Dar aceeași bucată de granit, dacă are șansa să treacă prin atelierul unui sculptor de talia lui *Michelangelo* devine altceva. Piatra nu mai există. Sau mai există, dar undeva departe într-un al n-lea plan (când $n \to \infty$). Ea a dispărut și a lăsat în locul ei ceva de la care oamenii cu greu își mai pot desprinde privirea.

Toți suntem alcătuiți din substanțe chimice (chestiile acelea care miros urât într-un laborator). Dar rezultatul? Vedeți cumva substanțe chimice când vă uitați în ochii cuiva? Simțiți gust de meta-clor-pirimidic-hexoză când sărutați? Toate astea au dispărut, sau mai precis s-au metamorfozat. E drept că ceva vă arde când atingeți o mână care vă este dragă, dar vă spun sigur că nu e acid sulfuric!

Un peisaj din natură: frunze căzute și intrate în stare de putrefacție, copaci mâncați de viermi, zăpadă plină cu excremente de animale. Nimic mai grețos. Dar privit de la distanță, e cu totul altceva.

Tudor Arghezi are o poezie din care citâm două versuri:

"Din bube, mucegaiuri și noroi
Iscat-am frumusețisi prețuri noi."

Toate astea le-am enumerat doar pentru ca să vă arătăm că lucruri aparent fără nici o valoare emoțională se pot transforma în ceva sublim.

Multe din problemele întâlnite în viața de zi cu zi pot fi exprimate cu ajutorul algoritmilor. Vom arăta cum putem exprima prin intermediul unui program mai mult decât rezolvarea unei probleme de teoria grafurilor, programare dinamică, backtracking sau branch and bound.

Un limbaj de programare îi lasă pe majoritatea oamenilor indiferenți. Toți spun că poate fi folosit pentru a calcula integrale, pentru a simula explozii nucleare, pentru a trasa grafice de bursă. Și cam atât (spun majoritatea). Dar dacă... O mică poveste ar merita să fie spusă aici:

Trăia odată un împărat care primea aproape în fiecare zi de la un cerșetor câte un măr. Împăratul lua mărul cu un gest indiferent și îl dădea unui slujitor care apoi îl arunca într-o groapă. Într-o zi în sala tronului a intrat o maimuță. Împăratul a luat mărul pe care tocmai îl primise și-l dădu maimuței. Aceasta mușcă mărul cu poftă și din el căzu un diamant...

Vrem să spunem - spre exemplu - că se poate face o declarație de dragoste sub forma unui program *Pascal*. Să spui cuiva *"Te iubesc!"* e foarte simplu. Cuvântul acesta este cel mai ieftin dintre toate. Se învață repede, nu costă nimic, se pronunță ușor, e plăcut... și e cel mai lipsit de imaginație. Și ***dragostea*** înseamnă în primul rând imaginație!

Dar într-un program *Pascal* putem ascunde aproape orice: o poveste sau o rețetă de mâncare sau un meci de fotbal...

Dar nimeni nu ne împiedică să facem și alte lucruri.

Cel mai des am folosit instrucțiunile `New` și `Dispose`. Aici am folosit `New` pentru a da naștere la ceva: la o dragoste, la un fiu de împărat, la un gol, și `Dispose` pentru a elimina sau distruge ceva: o dragoste, un zmeu, un jucător de pe teren etc. Pentru o mai bună înțelegere codurile sursă sunt comentate în detaliu.

Totodată, codurile sursă prezentate în acest articol demonstrează utilitatea uneia dintre axiomele din programare, și anume *importanța de a denumi sugestiv variabilele și procedurile dintr-un program.*

Un cod sursă care conține doar variabile de tipul `i`, `j`, `k`, `i1`, `i2`, `a`, `m`, `n` și procedurile `a`, `ap`, `c` nu spune nimic. Nu spune nici măcar ce problemă rezolvă.

Un program care conține variabilele și procedurile: Calculeaza_cel_mai_scurt_drum, Matricea_Drumurilor, Reconstituie_Drum spune aproape tot.

Nu "triumfați" când descoperiți că programele care urmează nu pot fi compilate. Din dorința de a exprima cu ajutorul lor mai mult decât niște simple idei informatice am fost nevoiți ca pe alocuri să forțăm nota.

Listing POVESTE.PAS

```
Program Poveste;
    { adaptare dupa basmele romanesti }

const Loc_Imparat=1;
          {          regiunea in care }
          { locuieste fiul de imparat  }
      Loc_Zmeu = 100;
                    { regiunea in care }
                    { locuieste zmeul  }
      fiu_de_imparat1 = 0;
                          { primul fiu }
          { de imparat era o nulitate }
      fiu_de_imparat2 = 0;
                         { al doilea }
                    { fiu era la fel }

type TDrum = set of Locuri;
   { un drum este o multime de locuri }
   { prin care se plimba o fiinta vie }
     OM = set of calitati;
        { un om este reprezentat ca o }
        {      multime de calitati    }

var drum_fiu1,drum_fiu2,drum_Praslea:
                                TDrum;
    Praslea:^OM;
          { Praslea era un om dinamic }
          { cu foarte multe calitati  }

procedure Drumuri_Minime_Intre_Regiuni;
{   Aceasta procedura calculeaza cele }
{    mai scurte drumuri intre oricare }
{ doua regiuni cunoscute sau necunos- }
{  cute de pe vremea imparatului Rosu }

var i,j,k:Byte;

begin
  for k:=1 to n do
    for i:=1 to n do
      if a[i,k]>0 then
        for j := 1 to n do
          if a[k,j]>0 then
            if a[i,k]+a[k,j]>a[i,j] then
              a[i,j]:=a[i,k]+a[k,j]
end;
```

```
procedure Determina_Drum_Zmeu(var
                    drum_zmeu : TDrum);
begin
  drum_zmeu:=[Loc_Zmeu]+[La_Vanatoare]
          { drumul zilnic al zmeului }
          {    este de acasa pana la }
          {      locul unde vaneaza  }
end;

procedure Determina_Drum_Fiu_De_Imparat(
                  var drum_fiu:TDrum);
          { determina pe unde merge }
          {      un fiu de imparat  }
begin
  while drum_fiu*drum_zmeu<>[] do
          { cat timp fiul de imparat }
          { nu se intalneste cu zmeul }
  drum_fiu := drum_fiu + [Random(100)]
            { fiul de imparat alege }
            { aleator o alta regiune }
            {      in care sa mearga }
end;

procedure Determina_Drum_Praslea(
                var drum_Praslea:TDrum);
begin
            {    Praslea calculeaza cu }
            { algoritmul Roy-Warshall }
            { cel mai scurt drum intre }
            {      oricare doua puncte }
            {             de pe harta  }
  Drumuri_Minime_Intre_Regiuni;
                  { apoi reconstituie }
                  { cel mai scurt drum }
                  { intre regiunea lui }
                  {  si cea a zmeului  }
  drum_Praslea:=
   Reconstituie_Cel_Mai_Scurt_Drum_Intre
                (Loc_Imparat,Loc_Zmeu)
           { Reconstituirea drumului }
           {             este lasata }
           {  in seama cititorului!  }
end;

Begin              { inceputul povestii }
{ comentariile apartin povestitorului! }
  SetTime(1207,februarie,7,Rosu);
  { momentul inceperii actiunii este }
  { anul 1207, luna februarie, ziua 7 }
  {       pe vremea imparatului Rosu }
  New(fiu_de_imparat1);
     { se naste primul fiu de imparat }
  fiu_de_imparat.Nume:='NoName';
          { nu are importanta numele }
          {  primului fiu de imparat }
```

```
New(fiu_de_imparat2);
{ se naste al doilea fiu de imparat }
fiu_de_imparat.Nume:='NoName';
{ nu are importanta numele celui }
{ de al doilea fiu de imparat }
New(Praslea);
{ se naste Praslea, cel de-al }
{ treilea fiu de imparat }
Determina_Drum_Zmeu(drum_zmeu);
{ determina drumul zmeului }
Determina_Drum_Fiu_De_Imparat
(drum_fiu1);
{ determina drumul primului }
{ fiu de imparat }
if drum_zmeu*drum_fiu1<>[] then
{ daca drumul zmeului }
{ se intersecteaza }
{ cu cel al }
{ primului fiu de imparat }
begin
New(lupta);
lupta^.durata = 1;
{ lupta dureaza }
{ o milisecunda }
Dispose(fiu_de_imparat1)
{ fiul de imparat este ucis }
end;
Determina_Drum_Fiu_De_Imparat
(drum_fiu2);
{ determina drumul }
{ celui de-al doilea }
{ fiu de imparat }
if drum_zmeu*drum_fiu2<>[] then
{ daca drumul zmeului }
{ se intersecteaza cu }
{ cel al celui de-al }
{ doilea fiu de imparat }
begin
New(lupta);
lupta^.durata = 1;
{ lupta dureaza }
{ o milisecunda }
Dispose(fiu_de_imparat1)
{ fiul de imparat este ucis }
end;
Determina_Drum_Praslea(drum_Praslea);
if drum_zmeu*drum_Praslea<>[] then
{ daca drumul zmeului }
{ se intersecteaza cu }
{ cu cel al lui Praslea }
begin
New(lupta);
lupta^.durata=24*60*60*1000;
{ prima parte a luptei }
{ tine o zi }
Delay(300000);
{ cei doi sunt obositi }
{ si fac o pauza de 5 minute }
New(lupta);
lupta^.durata=24*60*60*1000;
{ a doua parte a luptei }
{ tine o zi }
Delay(300000);
{ cei doi sunt obositi }
{ si fac o pauza de 5 minute }
Add(Praslea,apa);
{ Praslea bea un cioc de apa }
Dispose(zmeu)
{ zmeul este ucis }
end;
New(Casatorie);
Casatorie^.mire:='Praslea';
Casatorie^.mireasa:=
'Ileana Cosanzeaza';
Halt(0); { povestea se termina cu }
{ bine pentru Praslea }
End. { sfarsitul povestii }
```

Listing MECI.PAS

```
Program MeciFotbal;
{ un ipotetic meci de fotbal }
{ intre Romania si Italia }

uses { se vor folosi }
CartonaseGalbene,
{ cartonase galbene }
CartonaseRosii,
{ cartonase rosii }
SticleIncendiare,
{ sticle incendiare }
11_Jucatori_Romani,
11_Jucatori_Italieni,
3_Rezerve_Romani,
2_Rezerve_Italieni,
3_Arbitrii;

var balon:string;
{ jucatorul aflat in }
{ posesia balonului; }
{ balonul poate fi }
{ la unul dintre }
{ jucatori in aut }
{ sau in poarta }

procedure FluierArbitru;
begin
Sound(200);
Delay(200);
Nosound
end;
```

```
function FluierFinal:Boolean;
begin
  if minut=90+Minute_Prelungiri then
    begin
      FluierFinal:=true;
      FluierArbitru;
      FluierArbitru;
      FluierArbitru;
       { la finalul meciului arbitrul }
       {          fluiera de trei ori }
    end
  else FluierFinal:=false
end;

function FaultGrav(fault:TFault):
                                Boolean;
begin
  FaultGrav:=Pozitie(fault^.pecine)=
                              'La_Pamant'
            {       faultul este grav }
            {          daca jucatorul }
            {  faultat este la pamant }
end;

Begin
  minut:=0;
  Minute_Prelungiri:=0;
  FluierArbitru;
                {   Arbitrul fluiera }
                { inceputul meciului }
  while not FluierFinal do
                     { cat timp nu s-a }
                     { incheiat meciul }
  begin
    balon:='Hagi';
           { balonul se afla la Hagi }
    balon:='Popescu';
             { pasa la Giga Popescu }
    balon:='Adrian Ilie';
      { balonul ajunge la Adrian Ilie }
    balon:='aut';
              { mingea iese in aut }
    minut:=minut+1;
            { a mai trecut un minut }
    if minut=45
      then Delay(900000);
              {   pauza de 15 minute }
              { Hagi marcheza un gol }
    New(gol);
    gol^.autor:='Hagi';
           { Vladoiu il faulteaza pe }
           {             Roberto Bagio }
    New(fault);
    fault^.cine:='Vladoiu';
    fault^.pecine:='Bagio';
    if FaultGrav(fault) then
      begin
        Inc(Nr_Cartonase_Rosii);
        { la multimea cartonaselor lui}
        {       Vladoiu se mai adauga }
        {                  inca unul }
        Include(Vladoiu.cartonase,
                          Cartonas_Rosu;
        Dispose(Vladoiu);
             { Vladoiu este eliminat }
        Inc(Minute_Prelungiri)
             {     se mareste numarul }
             { minutelor de prelungire}
      end
  end
End.
```

Listing RETETA.PAS

```
Program Reteta_Pizza;

uses sare,piper,faina,apa,branza,sunca;
             { ingredientele folosite }

const Timp_Stat_InCuptor=10*60*1000;
   { in cuptor pizza va sta 10 minute }

procedure Initializeaza_Cantitati;
begin
                { cantitatile in care }
                { se vor utiliza unele }
                {          ingrediente }
                {     descrise mai sus }
                {   unitatea de masura }
                {          este gramul }
  Cantitate_Faina:=300;
  Cantitate_Branza:=100;
  Cantitate_Sunca:=50;
  Cantitate_Sare:=3
end;

Begin
  Initializeaza_Cantitati;
  Aluat:=[faina]+[apa];
  repeat
    Amesteca_Aluat
  until Aluat_Gata;
  Aluat:=Aluat+[sunca]+[branza];
                { pe aluat punem }
                { branza si sunca }
  cuptor:=pizza;
             { punem pizza in cuptor }
  Delay(600000)
    { tinem pizza 10 minute in cuptor }
End.
```

Listing LOVE.PAS

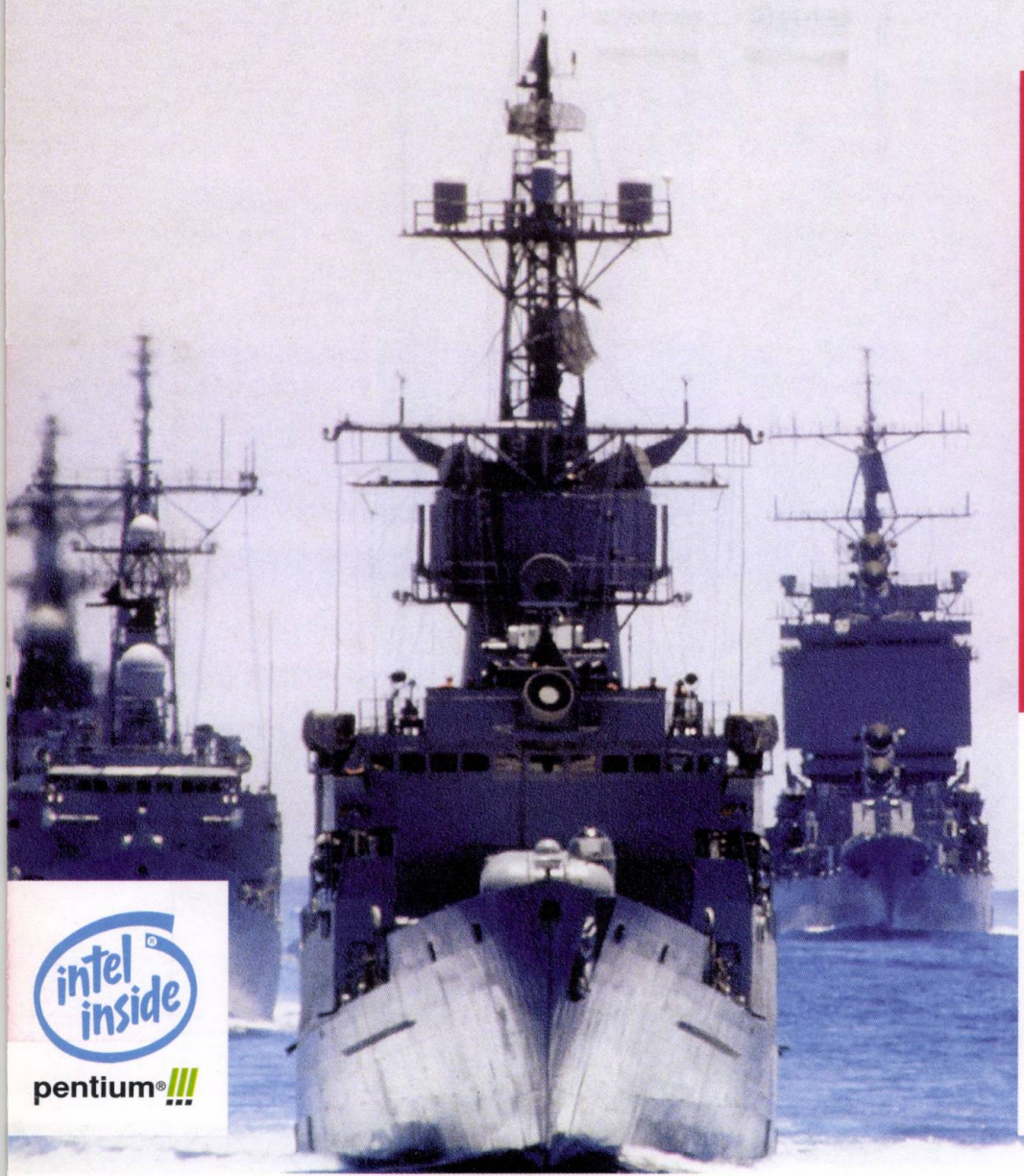
```
Program Love_Story;
uses                { cei doi au folosit }
  biletele_de_dragoste,
  saruturi,
  declaratii_de_dragoste,
  lacrimi;

Begin
                    {       aici a inceput }
                    { povestea de dragoste }
  Settime(primavara);
                              { era primavara }
  New(DRAGOSTE);
       { s-a infiripat o noua poveste }
       {   de dragoste intre EL si EA }
  if EL*EA<>[] then
                      { cand se intalneau }
    begin
                                { se sarutau }
      Sarut(EL,EA);
      Sarut(EA,EL);
          { stateau dublu inlantuiti }
      EL^.Urm:=EA;
      EA^.Pred:=EL
    end
  else            { cand nu se intalneau }
    begin
                    { isi trimiteau mesaje }
      SendMail(From:EL@o.iubesc.pe.ea,
                To:EA@il.iubesc.pe.el);
       SendMail(From:EA@il.iubesc.pe.el,
                 To:El@o.iubesc.pe.ea)
    end;
            { dragostea lor crestea in }
            {             fiecare clipa }
  clipa:=clipa+1;
  Dragostea_Lor:=Dragostea_Lor+1;
  Randomize;
     { s-a initializat generatorul de }
     {           evenimente aleatoare }
  factor1:=Random;
  factor2:=Random;
       { s-au generat doua evenimente }
       {    aleatoare independente de }
       {                 vointa lor }
  DRAGOSTE:=DRAGOSTE+[factor1]+
                                [factor2];
     { in dragostea lor au intervenit }
     {        cativa factori aleatori }
  Dispose(DRAGOSTE);
  { din cauza acestor factori cei doi }
  {    au fost nevoiti sa se desparta }
End.                { aici s-a terminat }
                    { povestea de dragoste }
```

Listing DECLARATIE.PAS

```
Program Declaratie_De_Dragoste;
const Dragostea_Mea_Pentru_Tine;
      {       dragostea mea pentru tine }
      { nu are sa se schimbe niciodata }

function Traiesc:Boolean;
                {       returneaza true }
                { daca eu sunt in viata }
begin
  Traiesc:=varsta<100
        { voi trai pana la 100 de ani }
end;

procedure Te_Iubesc;
              {       aici vom descrie }
              { ce inseamna sa iubesti }
begin
  SendMessage('Te iubesc!');
           { iti voi spune Te iubesc! }
  ShowLove
           { iti voi arata dragostea }
end;

Begin                          { declaratia }
  Dragostea_Mea:=0;
{ la inceput nu stiam ce este iubirea }
  repeat
    Find_Next_Girl;
  until Viata_Mea*Viata_Ta<>[];
   { chiar daca am fost cu multe fete }
   {    pana in clipa in care vietile }
   {          noastre s-au intalnit }
  if Viata_Mea*Viata_Ta=[] then
  { daca nu te-as fi intalnit pe tine }
    Viata_Mea:=nil;
  {    viata mea ar fi fost fara sens }
  Sufletului_Meu:=@TU;
   {    TU esti adresa sufletului meu }
  clipa:=clipa+1;
  Dragostea_Mea:=Dragostea_Mea+1;
          { dragostea mea pentru tine }
          {   creste in fiecare clipa }
  with Tine do        { Cu tine       }
    Vreau_Sa_Raman; { vreau sa raman }
  for Tine to Infinit downto
    Soarele_De_Pe_Cer;
       {      pentru tine la nesfarsit }
       { voi cobori soarele de pe cer }
  while Traiesc do Te_Iubesc
      { cat timp voi trai te voi iubi }
End.              { sfarsitul declaratiei }
```

Mihai Oltean și Crina Groșan sunt doctoranzi ai Universității Babeș-Bolyai din Cluj. Pot fi contactați prin e-mail la molteanu@cs.ubbcluj.ro, *respectiv* cgrosan@cs.ubbcluj.ro.

Acum ai propria ta forță de reacție rapidă!



Acum poți să faci față oricăror situații neprevăzute. Acum ești înarmat cum trebuie. Noul Armada M300 cu procesor mobil Intel® Pentium® III 600 MHz cu tehnologie SpeedStep™ îți dă toată puterea de foc necesară: display de până la 12,1"; hard-disc 12 GB; numai 2,2 cm grosime; modem 56K Mini-PCI integrat. Unitatea Mobilă de Expansiune (opțional) înseamnă economie de spațiu și rază mare de acțiune, prin transformarea noului Armada M300 într-o adevărată stație de luptă: acces la încă o baterie, încă un hard disc, la CD-ROM (sau DVD), unitate de dischetă etc.
Cu noul Armada M300 de partea ta, războiul e ca și câștigat... pentru că se termină înainte de a începe.

intel inside
pentium® III

ntel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

NonStop™

net consulting

Str. Știrbei Vodă 15, București, România
Tel: 223 27 01, 223 26 80; Fax: 312 41 99
E-mail: sales@netconsulting.ro
www.netconsulting.ro

Din păcate, majoritatea consideră că o relație la birou este nepotrivită.

Iar motivul cel mai des invocat este c
ar fi contra-productivă. În realitat
nimic mai fals!
Cu noul Compaq Deskpro EN Slir
Desktop cu procesor Intel® Pentium®
866 MHz poți stabili o relație extrer
de productivă încă din prima zi. Pentr
că este un sistem compact, prietenc
și adaptabil, care nu-ți va cre
niciodată vreo problemă.
Deskpro EN Slim Desktop concentrea
tehnologie de vârf într-un spațiu c
20% mai mic decât al oricărui alt P
hard-disc silențios Ultra ATA-100 c
până la 20GB, unitate CD-ROM, 128 M
memorie RAM, adaptor Intel Fa
Ethernet integrat și multe altel
În concluzie, o legătură de durat
inteligentă, va fi oricând acceptată c
restul colegilor ca fiind firească.
Să fie acesta singurul motiv pentru ca
ești privit cu admirație...?

intel inside
pentium® III

Intel, logotipul Intel Inside și Pentium sunt mărci înregistrate ale Intel Corporation.

COMPAQ

NonStop™

OMNILOGIC

Șos. București - Ploiești 135
Tel: 303 31 00; Fax: 303 31 64
www.omnilogic.ro

Gfo

Special edition
CEOI 2000

gazeta de informatica

A Magazine for the Future IT Specialists

Look inside for:

ACTUAL
CEOI 2000 - Cluj-Napoca

CONTEST
The Agora Programming Contest

CEOI2000

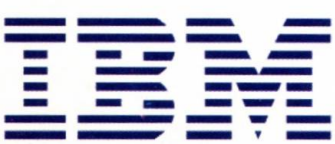
TEHNO
The processor WAR
INTEL vs. AMD

BABEL
SETI@home

FOCUS
Evaluation at CEOI 2000

INTERVIEW
Past & future Horia Georgescu
About CEOI in Cluj-Napoca Anca Voicu

sponsors of CEOI 2000



The Editorial Staff:

Scientific Coordinator:
Prof. dr. *Horia Georgescu (Bucharest)*
Editor in Chief:
Clara Ionescu (Cluj)
Senior Editor:
Mihai Scorțaru (Cluj)
Contributors:
Emanuela Cerchez (Iași)
Bogdan Dumitru (Ploiești)
Adrian Monea (Cluj)
Anca Voicu (Bucharest)

Assistant of the Editor in Chief:
Gabriela Bucșa

Desktop and Publishing:
Adrian Monea, Călin Borz
Corrections:
Georgeta Moldovan
Ada Ionescu
Răzvan Ristea
Simona Motogna
Cover and Graphics:
Andrei Pașa

Mail Address:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
E-mail address: office@ginfo.ro
Tel: 064/192422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro

ISSN: 1222-7129
Printed by: *Master Druck* Târgu Mureș
tel: 065/162.359
Editor: Computer Press Agora SRL
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș.
General Director:
Romulus Maier
Executive Director:
Adrian Pop
Technical Director:
Szabó László
Advertising:
Adrian Alexandru (aalex@agora.ro)
Bogdan Sârbu (bsarbu@agora.ro)
Ioana Udroiu (iudroiu@agora.ro)
Bucharest Subsidiary:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

gazeta de informatică

Summary

actual.. PAGE 3

CEOI 2000 - Cluj-Napoca

Croatia, The Czech Republic, Germany, Hungary, Poland, Romania, Slovenia, Slovakia, Moldova, The Netherlands, The United States of America

tasks.. PAGE 15

CEOI '98

CEOI '99

BOI 2000

solutions.. PAGE 27

CEOI '98
Zadar, Croatia

CEOI '99
Brno, Czech Republic

interview.. PAGE 34

PAST & FUTURE
HORIA GEORGESCU

About CEOI in CLUJ-N.
ANCA VOICU

contest.. PAGE 36

The Agora Programming Contest

The Second Edition of the Programming Contest Oganized by GInfo

babel.. PAGE 39

SETI@home

Help the Search for ExtraTerrestrial Intelligence Staying Home

tehno.. PAGE 43

The Processors WAR (INTEL vs. AMD)

1 GHz and Beyond this Psychological Barrier

focus.. PAGE 48

Evaluation at CEOI 2000

About the Evaluation System Used at CEOI 2K

A Few Thoughts About...

CEOI in ROMANIA

The day when we are surrounded by the participants at the Central European Olympiad in Informatics has come. According to the decision of the CEOI Olympic Committee, this edition takes place this year in Romania. And this does not happen by chance. The decision that this year's edition take place in Romania has several explanations.

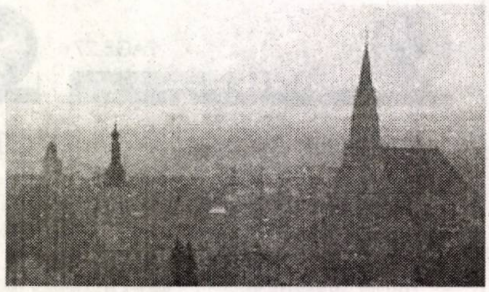
I sit in front of (no, not the white paper but) the white monitor and I search for appropriate words to express simply what I felt, many times, together with the Romanian participants while at other olympiads.

I think it all began in *Minsk*, at the first *International Olympiad in Informatics* in which the *Romanian* delegation, lead by *prof. dr. Horia Georgescu*, took part for the first time. The contestans of that edition were noticed, and *Romania*'s delegation had an important scientific contribution. The years passed and today *Romania* is rightfully considered a country that sends well-prepared contestants to these olympiads.

So the first reason would be the trust in the scientifical value of the Romanian informatics education system, which is widely recognized today. You may say that without the talent and the passion of the contestants nothing would be accomplished. You are right, but remember that in every garden, even the most beatiful flower would quickly wither and perish if it weren't watered by the gardner. But with daily care wonderful things can be obtained.

Returning to olympiads, this confidence grew over the years. In 1993, at *Romania*'s initiative, the first *Balkaniad in Informatics* was organized. Then, at the IOI held in *Argentina*, Radu Lupșa won an "absolute gold".

The next year saw the first edition of the CEOI, in *Romania*, at "*Tiberiu Popoviciu*" *Computer Science High-School*, in *Cluj-Napoca*. We will never forget the hard work done by the organizing and scientific committees so that our guests would go home with pleasant memories. But the true joy is felt today, seeing that this contest went on and "grew up" every year by the organizing countries. Since 1994, CEOI has become a tradition, organized continously, each year, in another city, by another country. Starting with 1997, the *United States*' team has participated almost every year as a guest, because the scientific standards are considered the highest of all informatics contests in the world.



Expectations oblige, especially considering *Romania*'s results in the years that followed. For example, we can not forget that in 1998, *Mihai Stroe* won another "absolute gold" at the *International Olympiad in Informatics*, and that the next year *Mihai Pătrașcu* and *Ștefan Radu* won two gold medals at the CEOI in *Brno*.

At the IOI *Romania* obtained good results, ranking among the first ten teams in the world. In 1998 *Romania* was asked to organize the IOI in 2007...

Meanwhile, the CEOI will take place in *Romania*. It has been a difficult job following the 1999 IOI edition organized perfectly in *Turkey*, at scientific and organizational standards never set before. *Romania*'s *Ministry of National Education* supported the idea of this contest from the beginning, but let's not forget that organizing such a contest at high standards imposed by previous editions takes a considerable financial effort from the sponsors. We thank them warmly. It is also worth noting that a lot of hard work and abnegation is required from all persons involved. We hope we'll be up to the job.

Like in Slovakia in 1996, the scientific activity of this edition is ensured mostly by students, all of them ex-olympics. We, their ex-teachers, were amazed by how quickly they surpassed us, how understanding they were and how enthusiastically they assumed high responsabities.

In the end, I wish all the contestants good luck. Remember that these are the "Olympic Games" and that participating is more important that winning. Do not be sad if the medal you dreamed of does not shine the way you hoped, or even if you will not step on the podium. I hope you do not forget the adventure called CEOI 2000!

Clara Ionescu

CEOI 2000 – Cluj-Napoca

CEOI 2000 takes place at the Transylvania Hotel in Cluj-Napoca. It is attended by the eight CEOI members: Croatia, Czech Republic, Germany, Hungary, Poland, Romania, Slovakia, Slovenia and three special guests: Moldova, The Netherlands and The United States of America. In what follows you can read a short presentation of this year's participants.

After six years the CEOI returns to *Cluj-Napoca*. In 1994 the *Computer Science High-School* was the host of 39 contestants from eight countries: *Croatia*, *Czech Republic*, *Hungary*, *Poland*, *Romania*, *Moldova*, *Turkey* and *Yugoslavia*. This year we have eight CEOI member countries and *Moldova*, *The Netherlands* and *The United States of America* participate as invited teams. A total of 51 contestants "fight" for the CEOI 2000 medals.

This 7th edition of CEOI takes place at the *Transylvania Hotel* because we wish to offer accomodation and professional activities in the same place. The more than 100 persons involved in organizing this event will, hopefully, make everything be OK.

Cluj-Napoca is a city in the middle of *Romania*, on the *Someș Mic River*. Formerly called *Cluj*, the city is the capital of *Cluj County*. It is an episcopal seat for the *Eastern Orthodox*, *Uniate*, and *Reformed* churches. Industries in the city include textile and paper mills, sugar refineries, breweries, and distilleries, and there are earthenware, soap, and candle factories.

Among the principal buildings of *Cluj-Napoca* are the *Church of St. Michael*, a Gothic structure built between 1350 and 1480; the house in which *Matthias Corvinus*, king of *Hungary*, was born in 1440; a *Reformed* church, dating back to 1386; the *Bánffy Palace* (today the *Art Museum*), residence of the former governor of *Transylvania* and the *University* building, constructed between 1893 and 1903.

Historians believe that *Cluj-Napoca* is built on the site of the ancient Roman settlement of *Napoca*. Saxons (Germans) settled there in 1178 and dominated the city until the 16th century. Becoming Lutherans after the *Reformation*, they had to leave the city following the establishment of the *Uniate* church during the *Counter-Reformation*. The city was the seat of government of *Transylvania* as a crownland of *Austria* and as a province of *Austro-Hungary*. In 1918 Transylvania united with the rest of Romania, union recognized officially in 1920. In 1940, during *World War II*, northern *Transylvania*, including *Cluj-Napoca*, was ceded to *Hungary*, but was liberated in 1945 at the end of the war. All this time *Cluj* was the center of important multicultural events. Along the centuries, in *Transylvania* many nationalities have lived together and contributed with their culture to create an exquisite and unique image of these places.

But returning to the present, between August 24th and 31st this city is the host of the *Central European Olympiad in Informatics*.

CEOI 2000 is organized by the *Romanian Ministry of National Education*, the *School Inspectorate of Cluj County*, the *Computer Press Agora* publishing house and the *Babeș-Bolyai University* in *Cluj-Napoca*. The *GInfo Editorial Office* also has an important role in organizing CEOI 2000.

We wish all the participants a pleasant stay in *Cluj-Napoca* and good luck at the competition.

In the following pages you may read several pieces of information about each participating country. You'll see the map and the official flag of the country, some general information and the name of medalists at the last two CEOI's.

The material was addapted from The Microsoft Encarta Encyclopedia.

Kako si!

CROATIA

Croatia is a republic in *South-Eastern Europe*, in the *Balkan Peninsula*, bounded on the North by *Slovenia* and *Hungary*, on the East and South by *Bosnia and Herzegovina* and on the East by *Serbia*. The area around the *Croatian* city of *Dubrovnik*, located at the southernmost tip of the republic's long western coastline on the *Adriatic Sea*, has a short border with *Montenegro*. Since 1991 *Serbia* and *Montenegro* have constituted the *Federal Republic of Yugoslavia*. Formerly a constituent republic of the *Socialist Federal Republic of Yugoslavia*, *Croatia* declared its independence on *June 25, 1991*. The country has an area of about 56,538 km². *Zagreb* is the republic's capital and largest city.

Croatia was the organizing country of the fifth edition of the CEOI. That edition took place in Zadar two years ago. The awards obtained by the Croatian team at the last two editions of CEOI were:

CEOI'98

Zvonimir Bujanovic, bronze medal
Zoran Majstrovic, bronze medal
Milan Mesic, bronze medal

Czech Republic CEOI 99 Brno, FI MU

Mario Zivic, bronze medal
Frane Saric, bronze medal

TKO RANO RANI
DVIJE SREĆE
GRABI

One who gets up early is twice as lucky!

Ahoj!

CZECH REPUBLIC

Czech Republic is a republic comprising the historic regions of *Bohemia*, *Moravia*, and part of *Silesia*, in *Central Europe*, bordered on the North by *Poland*, on the East by *Slovakia*, on the south by *Austria*, and on the West and North by *Germany*. Formerly parts of *Czechoslovakia*, the *Czech Republic* and *Slovakia* emerged as independent republics on *January 1, 1993*. The *Czech Republic* has an area of 78,864 km^2. *Prague* is the republic's capital and largest city. The *Czech Republic* is a parliamentary republic, governed under a constitution adopted in *December 1992* in the process of the division of the former *Czechoslovakia*. The government of the *Czech Republic* retained many of the elements of the government of *Czechoslovakia*.

The Czech Republic was the organizing country of the sixth edition of the CEOI. That edition took place in Brno one year ago. The awards obtained by the Czech Republic team at the last two editions of CEOI were:

No medals

Pavel Charvat, silver medal
Pavel Nejedly, silver medal
Zdenek Dvorak, bronze medal
Jiri Svoboda, bronze medal

DVAKRÁT MĚŘ, JEDNOU ŘEŽ.

Look before you leap!

Guten Tag!

GERMANY

The *Federal Republic of Germany* is a country in *Central Europe*, bordered on the North by the *North Sea*, *Denmark*, and the *Baltic Sea*, on the East by *Poland* and the *Czech Republic*, on the South by *Austria* and *Switzerland*; and on the West by *France*, *Luxembourg*, *Belgium*, and the *Netherlands*. A unified nation for 74 years (1871-1945), it was divided after *World War II* into the *Federal Republic of Germany* (FRG; commonly known as *West Germany*) and the *German Democratic Republic* (GDR; commonly known as *East Germany*). On *October 3rd, 1990*, *East Germany* became part of the FRG, and *Germany* once again became a unified nation. It has an area of 356,959 km²; Berlin is Germany's capital and largest city.

Germany has recently become a CEOI member, but hasn't organized a CEOI edition yet. The German team participated as guest team in the last two CEOI editions and obtained several awards:

CEOI'98

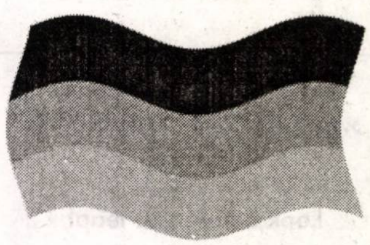
Jens Zumbragel, bronze medal
Simon Giesecke, bronze medal

Czech Republic CEOI 99 Brno, FI MU

Dominik Schultes, bronze medal

Wer den Pfennig
nicht ehrt,
ist des Talers
nicht wert.

Take care of the pence, and the pounds will take care of themselves!



Szia!

HUNGARY

Hungary is a republic in *Central Europe*, bordered on the North by *Slovakia*; on the North-East by *Ukraine*, on the East by *Romania*; on the South by *Serbia* (part of the federation of *Serbia* and *Montenegro*), *Croatia*, and *Slovenia* and on the West by *Austria*. The total area of *Hungary* is 93,030 km^2. The republic's capital and largest city is *Budapest*. The centre of tourism in *Hungary* is *Lake Balaton*. It is the largest lake in *Hungary* having an area of about 673 km^2. It is fed by more than 30 streams, the chief of which is the *Zala*, and is drained by the *Sarviz*, a tributary of the *Danube*. This is the largest freshwater lake in *Central Europe*.

Hungary was the organizing country of the second edition of the CEOI. That edition took place in Szeged five years ago. The awards obtained by the Hungarian team at the last two editions of CEOI were:

CEOI'98

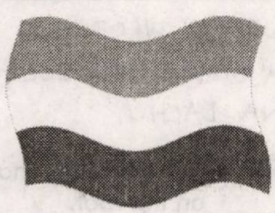
Andras Forhecz, silver medal
Andras Rocob, silver medal
Zsolt Felfoldi, bronze medal
Balazs Racz, bronze medal

Czech Republic CEOI 99 Brno, FI MU

Andras Rokob, silver medal
Adam Novak, bronze medal

Bagoly mondja verébnek hogy nagy fejü.

The owl tells the sparrow that he has a big head!



Dzień dobry!

POLAND

The *Republic of Poland* is a country in *Central Europe*, bordered on the North by the *Baltic Sea* and *Russia*, on the East by *Lithuania*, *Belarus* and *Ukraine*, on the South by the *Czech Republic* and *Slovakia* and on the West by *Germany*. The area of the country is 312,677 km^2. The republic's capital and largest city is *Warsaw*. *Poland* was one of the major *European* powers under the *Jagiellon* dynasty. With the end of the dynasty in 1572, *Poland* entered a long period of decline, culminating in the partition of the country between *Russia*, *Austria*, and *Prussia* in 1772, 1793, and 1795. *Poland* was again established as a sovereign state after *World War I*. It was partitioned for a fourth time in 1939 by *Germany* and the Union of Soviet Socialist Republics

Poland was the organizing country of the fourth edition of the CEOI. That edition took place in Novy Sacz three years ago. The awards obtained by the Polish team at the last two editions of CEOI were:

CEOI'98

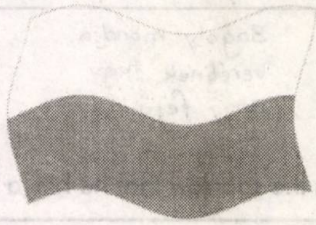
Andrzej Gasienica-Samek, gold medal
Eryk Kopczynski, gold medal
Tomasz Czajka, gold medal
Pawel Wolff, bronze medal

Czech Republic CEOI 99 Brno, FI MU

Andrzej Gasienica-Samek, gold medal
Krzysztof Onak, silver medal
Michal Nowakiewicz, bronze medal
Marcin Meinardi, bronze medal

LEPSZY WRÓBEL
W GARŚCI NIŻ GOŁĄB
NA DACHU.

A sparrow in the hand is better than a pigeon on the roof.



ROMANIA

Romania is a republic in *South-Eastern Europe*, bordered on the North by *Ukraine*, on the East by *Moldova*; on the South-East by the *Black Sea*, on the South by *Bulgaria*, on the South-West by *Serbia* (part of the federation of *Serbia* and *Montenegro*) and on the West by *Hungary*. The total area of *Romania* is about 237,500 km^2. *Bucharest* is *Romania*'s capital and largest city. From 1881 to 1947 Romania was a monarchy.In *February 1997*, *King Michael*, who had abdicated as monarch in 1947 and gone into exile, returned officially to Romania having a hero's welcome; *Emil Constantinescu*'s government (elected in 1996) discussed a possible return to a constitutional monarchy, but *Romania* remained a republic.

Romania was the country who initiated the Central European Olympiad in Informatics. The first edition of this contest took place in Cluj, six years ago. The awards obtained by the Romanian team at the last two editions of CEOI were:

CEOI'98

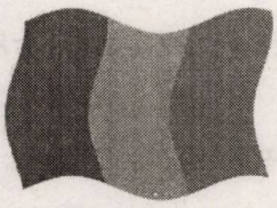
Mihai Scortaru, silver medal
Angel Proorocu, silver medal
Tudor Ioan Leu, bronze medal
Diodor Bitan, bronze medal

Czech Republic
CEOI 99
Brno, FI MU

Mihai Patrascu, gold medal
Stefan Radu, gold medal
Bogdan Dumitru, silver medal
Bogdan Batog, silver medal

Dacă n'ai
ce-ţi place
să-ţi placă
ce ai

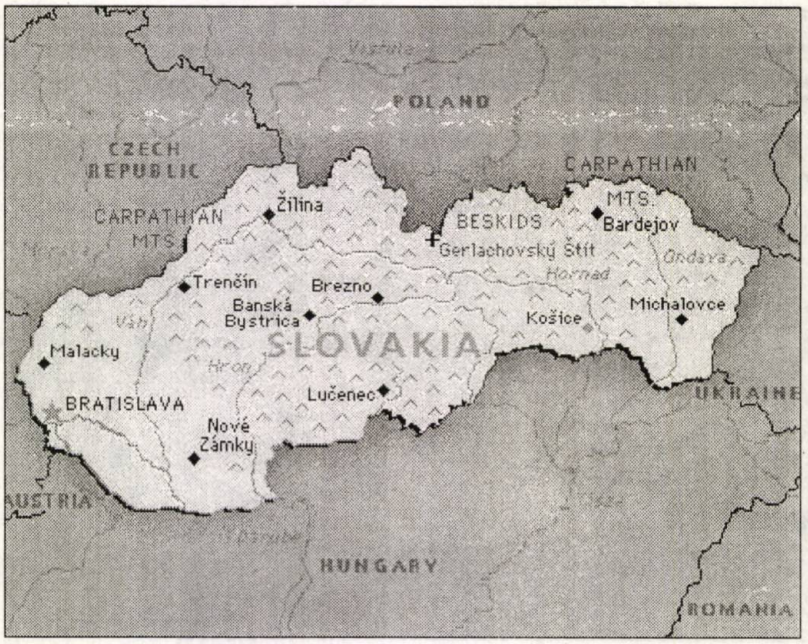
If you don't have what you like, you should like what you have!



Ahoj!

SLOVAKIA

Slovakia is a republic in *Central Europe*, bordered on the North by *Poland*, on the east by *Ukraine*, on the South by *Hungary* and on the West by *Austria* and the *Czech Republic*. Throughout the autumn of 1992, *Czechs* and *Slovaks* negotiated the details for disbanding the federal government. In *November 1992* the federal parliament of *Czechoslovakia* voted to dissolve the federation on *December 31, 1992*, despite polls indicating that a majority of citizens opposed to the breakup. Formerly parts of *Czechoslovakia*, *Slovakia* and the *Czech Republic* emerged as independent republics on *January 1, 1993*. *Slovakia* has an area of 49,035 km². *Bratislava* is the republic's capital and largest city.



Slovakia was the organizing country of the third edition of the CEOI. That edition took place in Bratislava four years ago. The awards obtained by the Slovakian Team at the last two editions of CEOI were:

CEOI'98

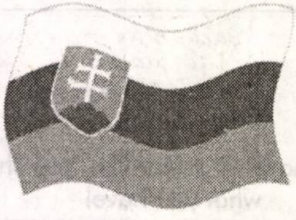
Jan Senko, gold medal
Jan Valky, silver medal
Michal Forisek, bronze medal

Czech Republic CEOI 99 Brno, FI MU

Jan Oravec, bronze medal
David Pal, bronze medal

DVAKRÁT MĚŘ, JEDNOU ŘEŽ.

Look before you leap!



SLOVENIA

Slovenia is a republic in *South-Eastern Europe*, on the *Balkan Peninsula*, bounded on the North by *Austria*, on the North-East by *Hungary*, on the South-East and South by *Croatia* and on the West by *Italy* and the *Adriatic Sea*. ***Slovenia***'s dissatisfaction with its membership of the *Socialist Federal Republic of Yugoslavia* grew during the 1980s, with increased sentiment first for greater autonomy and then for independence. Formerly a constituent republic of the *Socialist Federal Republic of Yugoslavia*, ***Slovenia*** proclaimed its independence in *June 1991*. It joined the *United Nations* in *May 1992*. The republic has an area of 20,254 km^2. *Ljubljana* is the capital and largest city.

Slovenia hasn't organized any CEOI editions yet and the Slovenian participants at the last two CEOI editions did not obtain any medals, so...

CEOI'98

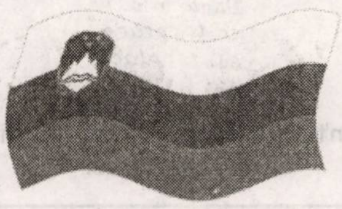
No medals.

Czech Republic CEOI 99 Brno, FI MU

No medals.

TKO RANO RANI
DVIJE SREĆE
GRABI

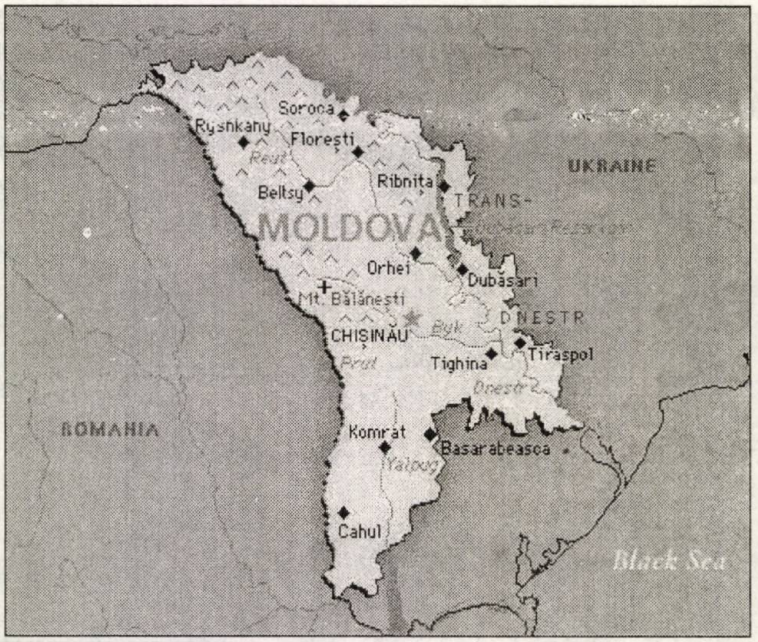
One that gets up early is twice as lucky!



Bună!

MOLDOVA

Moldova is a republic in *South-Eastern Europe*, bordered on the North, East, and South by *Ukraine* and on the West by *Romania*. It is also known as *Moldavia*, and was formerly the *Moldavian Soviet Socialist Republic* of the *Union of Soviet Socialist Republics*. The *Prut* River forms ***Moldova***'s western boundary, and the *Dnestr* River forms part of the country's North-Eastern and South-Eastern boundaries. *Chișinău*, the capital, is also the largest city. ***Moldova***'s total area is about 33,700 km². Present-day Moldova occupies the central two-thirds of a region historically referred to as *Bessarabia*. For centuries the name ***Moldova*** referred to a larger area encompassing *Bessarabia* and stretching from the *Black Sea* in the South to *Bukovina*, a former province of *Romania*, in the North, and from the *Siret River* in the west to the *Dnestr* in the East.



This is the second CEOI edition attended by Moldova as a guest team. The previous participation was at the first CEOI, which took place in Cluj. They did not take part in the last two CEOI editions, so...

CEOI'98

Not attended.

Czech Republic CEOI 99 Brno, FI MU

Not attended.

Dacă n'ai
ce-ți place
să-ți placă
ce ai

If you don't have what you like, you should like what you have!

THE NETHERLANDS

Netherlands, also known, unofficially, as *Holland* is a constitutional monarchy in *North-Western Europe*, bordered on the North and West by the *North Sea*, on the East by *Germany* and on the South by *Belgium*. With *Belgium* and *Luxembourg*, the *Netherlands* forms the *Low Countries* or *Benelux*. The *Netherlands Antilles* and *Aruba*, self-governing island territories in the *Caribbean*, are part of the kingdom. The *Netherlands* has a total area of 41,526 km^2. The country's capital and largest city is *Amsterdam*. The *Netherlands* is a constitutional and hereditary monarchy with a parliamentary system of government. The head of state of the *Netherlands* is the hereditary monarch, who has had little real power since the constitution was revised in 1848.

This year the Netherlands will participate for the second time at the Central European Olympiad in Informatics, as a guest team. They haven't attended the last two CEOI editions, so...

CEOI'98

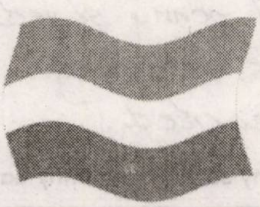
Not attended.

Czech Republic CEOI 99 Brno, FI MU

Not attended.

Wie goed doet,
goed ontmoet.

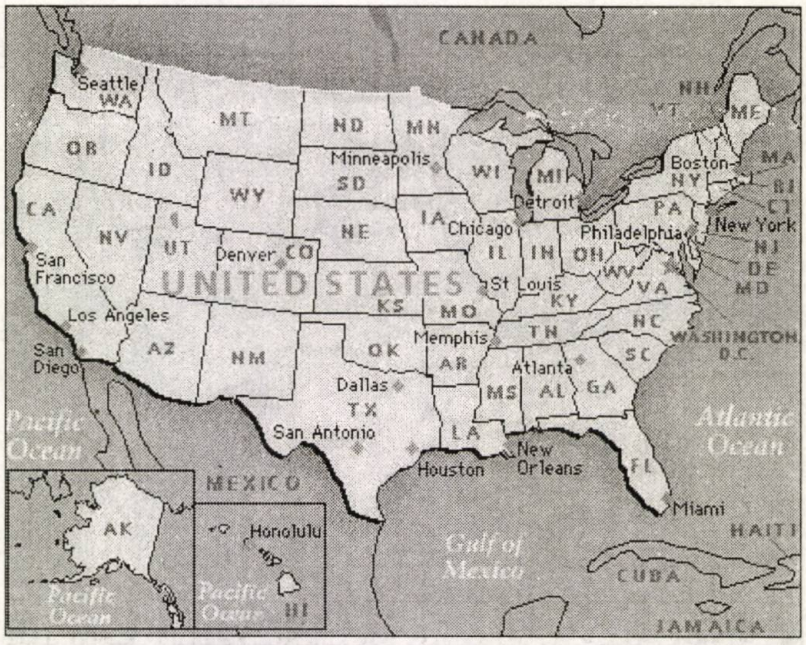
One who gives a lot, receives a lot!



Hello!

THE UNITED STATES OF AMERICA

United States of America, popularly referred to as the *United States* or as *America* is a federal republic on the continent of *North America*, consisting of 48 contiguous states and the non-contiguous states of *Alaska* and *Hawaii*. The conterminous 48 states are bounded on the North by *Canada*, on the East by the *Atlantic Ocean*, on the South by the *Gulf of Mexico* and *Mexico*, and on the West by the *Pacific Ocean*. The Northern boundary is partly formed by the *Great Lakes* and the *St Lawrence River*; the Southern boundary is partly formed by the *Rio Grande*. *New York* is the largest city in the *United States*; *Washington, D.C.* is the capital. The total area of the *United States* (including the *District of Columbia*) is 9,666,861 km².



The United States of America will participate at CEOI 2000 as a guest team. They have also been guests at CEOI '99 in Brno and obtained the following awards:

CEOI'98

Not attended.

Czech Republic CEOI 99 Brno, FI MU

Daniel Wright, gold medal
Benjamin Mathews, silver medal
Percy Liang, bronze medal

A penny saved is a penny earned.

A penny saved is a penny earned!

CEOI '98

CEOI '98 took place in Zadar, Croatia and was attended by participants from Bosnia and Hertzegovina, Croatia, Czech Republic, Germany, Hungary, Poland, Romania, Slovakia and Slovenia. Here you are presented the tasks the contestants had to solve during the two competition days.

CEOI9801: Squares

We are given N squares in the coordinate plane whose sides are parallel to the coordinate axes. All the corners have integer coordinates and the squares do not touch or overlap. You are required to count the number of squares visible from the origin point O, O=(0,0). A square is visible from the origin point O if there are two distinct points A and B on one of its sides such that the interior of the triangle OAB has no common points with any of the remaining squares.

Input data

The first line of the input file **SQUARES.IN** contains the integer N, $1 \le N \le 1000$, the number of squares.

Each of the following N lines describes a square by specifying integers X, Y and L separated by single blank characters, $1 \le X, Y, L \le 10000$. X and Y are coordinates of the lower left corner (the corner with the least X and Y coordinates) and L is the side length.

Output data

The first and the only line of the output file **SQUARES.OUT** should contain the number of squares that are visible from the origin.

Examples

SQUARES.IN

```
3
2 6 3
1 4 1
3 4 1
```

SQUARES.OUT

```
3
```

SQUARES.IN

```
4
1 2 1
3 1 1
2 4 2
3 7 1
```

SQUARES.OUT

```
2
```

CEOI9802: Cards

Alice and Bob have a set of N cards labelled with numbers 1...N (so that no two cards have the same label) and a shuffle machine. We assume that N is an odd integer. The shuffle machine accepts the set of cards arranged in an arbitrary order and performs the following operation of double shuffle: for all positions i, $1 \le i \le N$, if the card at the position i is j and the card at the position j is k, then after the completion of the operation of double shuffle, position i will hold the card k. Alice and Bob play a game. First Alice writes down all the numbers from 1 to N in some random order: $a_1, a_2, \ldots, a_N$. Then she arranges the cards so that the position a_i holds the card numbered a_{i+1}, for every $1 \le i \le N-1$, while the position a_N holds the card numbered a_1.

This way, the cards are put in some order $x_1, x_2, \ldots, x_N$, where x_i is the card at the ith position. Now she sequentially performs S double shuffles using the shuffle machine described above. After that, the cards are arranged in some final order $p_1, p_2, \ldots, p_N$ which Alice reveals to Bob, together with the number S. Bob's task is to guess the order $x_1, x_2, \ldots, x_N$ in which Alice originally put the cards just before giving them to the shuffle machine.

Input data

The first line of the input file **CARDS.IN** contains two integers separated by a single blank character: the odd integer N, $1 \le N \le 1000$, the number of cards, and the integer S, $1 \le S \le 1000$, the number of double shuffle operations.

The following N lines describe the final order of cards after all the double shuffles have been performed such that for each i, $1 \le i \le N$, the $(i+1)^{th}$ line of the input file contains p_i (the card at the position i after all double shuffles).

Output data

The output file **CARDS.OUT** should contain N lines which describe the order of cards just before they were given to the shuffle machine.

For each i, $1 \le i \le N$, the ith line of the output file should contain x_i (the card at the position i before the double shuffles).

Examples

CARDS.IN

```
5 2
4
1
5
3
2
```

CARDS.OUT

```
2
5
4
1
3
```

CARDS.IN

```
7 4
6
3
1
2
4
7
5
```

CARDS.OUT

```
4
7
5
6
1
2
3
```

CEOI9803: Substract

We are given a sequence of N positive integers $a=[a_1, a_2, \ldots, a_N]$ on which we can perform contraction operations. One contraction operation consists of replacing adjacent elements a_i and a_{i+1} by their difference $a_i - a_{i+1}$. For a sequence of N integers, we can perform exactly N-1 different contraction operations, each of which results in a new (N-1) element sequence.

Precisely, let con(a,i) denote the (N-1) element sequence obtained from $[a_1, a_2, \ldots, a_N]$ by replacing the elements ai and a_{i+1} by a single integer $a_i - a_{i+1}$:

$con(a,i) = [a_1, \ldots, a_{i-1}, a_i - a_{i+1}, a_{i+2}, \ldots, a_N]$

Applying N-1 contractions to any given sequence of N integers obviously yields a single integer.

For example, applying contractions 2, 3, 2 and 1 in that order to the sequence [12,10,4,3,5] yields 4, since:

```
con([12,10,4,3,5],2) = [12,6,3,5]
con([12,6,3,5]   ,3) = [12,6,-2]
con([12,6,-2]    ,2) = [12,8]
con([12,8]       ,1) = [4]
```

Given a sequence $a_1, a_2, \ldots, a_N$ and a target number T, the problem is to find a sequence of N-1 contractions that applied to the original sequence yields T.

Input data

The first line of the input file **SUBTRACT.IN** contains two integers separated by blank character: the integer N, $1 \le N \le 100$, the number of integers in the original sequence, and the target integer T, $-10000 \le T \le 10000$.

The following N lines contain the starting sequence: for each i, $1 \le i \le N$, the $(i+1)^{th}$ line of the input file contains integer a_i, $1 \le a_i \le 100$.

Output data

Output file **SUBTRACT.OUT** should contain N-1 lines, describing a sequence of contractions that transforms the original sequence into a single element sequence containing only number T. The ith line of the output file should contain a single integer denoting the ith contraction to be applied.

You can assume that at least one such sequence of contractions will exist for a given input.

Examples

SUBTRACT.IN

```
4 5
10
2
5
2
```

SUBTRACT.OUT

```
1
2
1
```

SUBTRACT.IN

```
5 4
12
10
4
3
5
```

SUBTRACT.OUT

```
2
3
2
```

CEOI9804: Soldiers

N soldiers of the land *Gridland* are randomly scattered around the country.

A position in *Gridland* is given by a pair `(x,y)` of integer coordinates. Soldiers can move - in one move, one soldier can go one unit up, down, left or right (hence, he can change either his `x` or his `y` coordinate by `1` or `-1`).

The soldiers want to get into a horizontal line next to each other (so that their final positions are `(x,y)`, `(x+1,y)`, ..., `(x+N-1,y)`, for some `x` and `y`). Integers `x` and `y`, as well as the final order of soldiers along the horizontal line is arbitrary.

The goal is to minimize the total number of moves of all the soldiers that takes them into such configuration.Two or more soldiers must never occupy the same position at the same time.

Input data

The first line of the input file **SOLDIERS.IN** contains the integer `N`, $1 \le N \le 10000$, the number of soldiers.

The following `N` lines of the input file contain initial positions of the soldiers: for each `i`, $1 \le i \le N$, the $(i+1)^{th}$ line of the input file contains a pair of integers `x[i]` and `y[i]` separated by a single blank character, representing the coordinates of the ith soldier, $-10000 \le x[i], y[i] \le 10000$.

Output data

The first and the only line of the output file **SOLDIERS.OUT** should contain the minimum total number of moves that takes the soldiers into a horizontal line next to each other.

Examples

SOLDIERS.IN	SOLDIERS.OUT
3 1 0 2 4 3 2	4

SOLDIERS.IN	SOLDIERS.OUT
5 1 2 2 2 1 3 3 -2 3 3	8

CEOI9805: Roads

`N` cities named with numbers `1...N` are connected by one-way roads. Each road has two parameters associated with it: the road length and the toll that needs to be paid for the road (expressed in the number of coins).

Bob and Alice used to live in the city `1`. After noticing that Alice was cheating in the card game they liked to play, Bob broke up with her and decided to move away - to the city `N`. He wants to get there as quickly as possible, but he is short on cash.

We want to help Bob to find the shortest path from the city `1` to the city `N` that he can afford with the amount of money he has.

Input data

The first line of the input file **ROADS.IN** contains the integer `K`, $0 \le K \le 10000$, maximum number of coins that Bob can spend on his way.

The second line contains the integer `N`, $2 \le N \le 100$, the total number of cities.

The third line contains the integer `R`, $1 \le R \le 10000$, the total number of roads.

Each of the following `R` lines describes one road by specifying integers `S`, `D`, `L` and `T` separated by single blank characters :

- `S` is the source city, $1 \le S \le N$
- `D` is the destination city, $1 \le D \le N$
- `L` is the road length, $1 \le L \le 100$
- `T` is the toll (expressed in the number of coins), $0 \le T \le 100$

Notice that different roads may have the same source and destination cities.

Output data

The first and the only line of the output file **ROADS.OUT** should contain the total length of the shortest path from the city 1 to the city `N` whose total toll is less than or equal to `K` coins. If such a path does not exist, only number -1 should be written to the output file.

Examples

ROADS.IN	ROADS.OUT
5 6 7 1 2 2 3 2 4 3 3 3 4 2 4 1 3 4 1 4 6 2 1 3 5 2 0 5 4 3 2	11

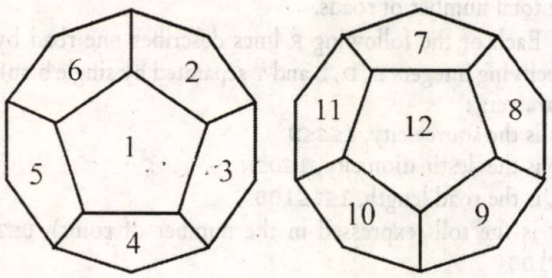
ROADS.IN	ROADS.OUT
0 4 4 1 4 5 2 1 2 1 0 2 3 1 1 3 4 1 0	-1

CEOI9806: Ball

Professor *Balltazar* is a big football fan. His birthday was just a couple of days before he was going to leave for the *World Cup Football '98* in *France*, so his friends gave him as

a present a dodecahedron-shaped puzzle as an entertainment while watching the boring games.

The puzzle has 12 equal pentagon sides, labelled with numbers 1...12. The figure below shows the two hemispheres of the dodecahedron, together with the side labelling we will use in this problem. Hemispheres are "glued" together in such a way that the side 7 is adjacent to the sides 8, 12, 11, 2 and 6 (sides are adjacent if they share an edge). In particular, edges a and b on the left hemisphere will be glued to the edges a and b on the right hemisphere shown below.



In addition to that, there are 12 pentagon-shaped tiles, also labelled from 1 to 12. Every edge on each of the tiles is marked with a number from the set {0, 1, 2}. Each tile can be placed on each of the twelve sides in any of the 5 positions obtained by rotating the tile around its centre.

To solve the puzzle, we need to put each tile on some of the twelve dodecahedron sides in some position, so that every two adjacent tiles have their common edge marked with the same number.

Help Professor Balltazar to solve the puzzle !

Input data

The input file **BALL.IN** contains 12 lines. For each i, $1 \le i \le 12$, the ith line describes the ith tile by specifying 5 numbers from the set {0,1,2} separated by single blank characters. This sequence shows the edge marking of the ith tile starting from an arbitrary edge (called the ith reference edge) going in the clockwise direction.

Output data

Output file **BALL.OUT** should contain the description of a solved puzzle in 12 lines with 2 integers in each line. For each i, $1 \le i \le 12$, the ith line should contain integers t[i] and n[i] separated by a single blank character describing the tile and its position on the ith side:

The ith side will hold the tile labelled t[i].

The tile can be placed on the ith side in five different positions. The exact position is specified by n[i], which denotes the label of the adjacent side which is in the direction of the t[i]th reference edge (looking from the centre of the ith side). Precisely, the t[i]th reference edge is placed on the dodecahedron edge shared by the sides labelled i and n[i].

If the puzzle can not be solved, only number -1 should be written to the output file.

Examples

BALL.IN

```
0 0 1 1 2
0 2 1 0 1
2 0 1 0 1
0 0 1 2 1
0 2 1 1 2
2 0 1 2 1
0 2 1 2 1
2 2 1 0 1
1 2 2 0 0
0 2 1 0 2
0 2 1 2 0
2 0 1 2 0
```

BALL.OUT

```
1 2
3 7
12 4
7 9
9 1
11 8
8 2
4 6
5 4
2 12
6 3
10 7
```

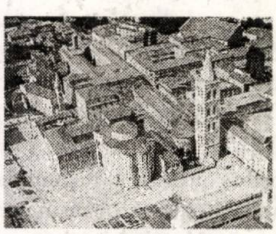
The Kornati Islands

BALL.IN

```
1 0 2 0 2
2 2 2 1 2
1 1 0 0 0
1 1 0 2 1
2 1 1 1 1
1 2 2 1 1
2 1 2 2 1
2 2 0 1 0
0 1 2 1 2
2 2 1 0 0
1 2 0 2 0
2 2 2 0 1
```

BALL.OUT

```
1 2
2 7
8 2
7 1
11 4
12 2
5 2
3 12
10 5
9 3
6 10
4 7
```



The city of Zadar

Brno, Czech Republic!

CEOI '99

CEOI '99 took place in Brno, Czech Republic and was attended by participants from Bosnia and Hertzegovina, Croatia, Czech Republic, Germany, Hungary, Poland, Romania, Slovakia, Slovenia and The United States of America. Here you are presented the tasks the contestants had to solve during the two competition days.

CEOI9901: A Game on the Chessboard

On the chessboard of size 4x4 there are 8 white and 8 black stones, i.e. one stone on each field. Such a configuration of stones is called a game position. Two stones are *adjacent* if they are on fields with a common side (i.e. they are adjacent in either horizontal or vertical direction but not diagonal).

This means that each stone has at most 4 neighbours. The only legal move in our game is exchanging any two adjacent stones. Your task is to find the shortest sequence of moves transforming a given starting game position into a given final game position.

Input

The starting game position is described in the first 4 lines of input file **GAME.IN**. There are 4 symbols in each line, which define the colour of each stone in the row from the left to the right. The lines describe the rows of the chessboard from the top to the bottom. Symbol '0' means a white stone and symbol '1' a black one. There is no space symbol separating the symbols in the line. The fifth line is empty. The next 4 following lines describe the final game position in the same way.

Output

The first line of output file **GAME.OUT** contains the number of the moves. The following lines describe the sequence of moves during the game. One line describes one move and it contains 4 positive integers R_1 C_1 R_2 C_2 separated by single spaces. These are the coordinates of the adjacent fields for the move, i.e. fields $[R_1, C_1]$ and $[R_2, C_2]$, where R_1 (or R_2) is the number of the row and C_1 (or C_2) is the number of the column. The rows on the chessboard are numbered from 1 (top row) to 4 (bottom row) and the columns are numbered from 1 (the leftmost column) to 4 (the rightmost one) as well (i.e. the coordinates of the left upper field are [1,1]). If there are multiple shortest sequences of moves transforming the starting position to the final position, you can output any one of them.

Example

GAME.IN

```
1111
0000
1110
0010

1010
0101
1010
0101
```

GAME.OUT (one of the solutions)

```
4
1 2 2 2
1 4 2 4
3 2 4 2
4 3 4 4
```

CEOI9902: Orders

The store's manager has sorted all kinds of goods in an alphabetical order of their labels. All the kinds having labels starting with the same letter are stored in the same warehouse (i.e. in the same building) labelled with this letter. During the day the store's manager receives and books the orders of goods which are to be delivered from the store. Each order requires only one kind of goods. The store's manager processes the requests in order of their booking.

You know in advance all the orders which will have to be processed by the store's manager today, but you do not know their booking order.

Compute all possible ways of the visits of warehouses for the store's manager to settle all the demands piece after piece during the day.

Input

Input file **ORDERS.IN** contains a single line with all labels of the requested goods (in random order). Each kind of goods is represented by the starting letter of its label. Only

small letters of the English alphabet are used. The number of orders doesn't exceed 200.

Output

Output file **ORDERS.OUT** will contain all possible orderings in which the store's manager may visit his warehouses. Every warehouse is represented by a single small letter of the English alphabet - the starting letter of the label of the goods. Each ordering of warehouses is written in the output file only once on a separate line and all the lines containing orderings have to be sorted in an alphabetical order (see the example). No output will exceed 2 megabytes.

Example

ORDERS.IN

```
bbjd
```

ORDERS.OUT

```
bbdj
bbjd
bdbj
bdjb
bjbd
bjdb
dbbj
dbjb
djbb
jbbd
jbdb
jdbb
```

CEOI9903: Parity game

Now and then you play the following game with your friend. Your friend writes down a sequence consisting of zeroes and ones. You choose a continuous subsequence (for example the subsequence from the third to the fifth digit inclusively) and ask him, whether this subsequence contains even or odd number of ones. Your friend answers your question and you can ask him about another subsequence and so on. Your task is to guess the entire sequence of numbers.

You suspect some of your friend's answers may not be correct and you want to convict him of falsehood. Thus you have decided to write a program to help you in this matter.

The program will receive a series of your questions together with the answers you have received from your friend. The aim of this program is to find the first answer which is provably wrong, i.e. that there exists a sequence satisfying answers to all the previous questions, but no such sequence satisfies this answer.

Input

The first line of input file **PARITY.IN** contains one number, which is the length of the sequence of zeroes and ones. This length is less or equal to 1000000000.

In the second line, there is one positive integer which is the number of questions asked and answers to these questions. The number of questions and answers is less or equal to 5000.

The remaining lines specify questions and answers. Each line contains one question and the answer to this question: two integers (the position of the first and last digit in the chosen subsequence) and one word which is either 'even' or 'odd' (the answer, i.e. the parity of the number of ones in the chosen subsequence, where 'even' means an even number of ones and 'odd' means an odd number).

Output

There is only one line in output file **PARITY.OUT** containing one integer X.

Number X says that there exists a sequence of zeroes and ones satisfying first X parity conditions, but there exists none satisfying X+1 conditions.

If there exists a sequence of zeroes and ones satisfying all the given conditions, then number X should be the number of all the questions asked.

Example 1

PARITY.IN

```
10
5
1 2 even
3 4 odd
5 6 even
1 6 even
7 10 odd
```

PARITY.OUT

```
3
```

Example 2

PARITY.IN

```
10
5
1 2 even
1 4 even
2 4 odd
1 10 even
3 10 even
```

PARITY.OUT

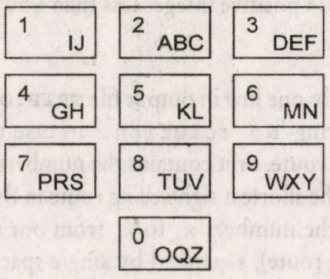
```
5
```

CEOI9904: Phone numbers

In the present world you frequently meet a lot of call numbers and they are going to be longer and longer. You need to remember such a kind of numbers. One method how to do it in an easy way is to assign letters to digits as shown in the picture.

This way every word or a group of words can be assigned a unique number, so you can remember words in-

stead of call numbers. It is evident that it has its own charm if it is possible to find some simple relationship between the word and the person itself. So you can learn that the call number 941837296 of a chess playing friend of yours can be read as WHITEPAWN, and the call number 2855304 of your favourite teacher is read BULLDOG.



Write a program to find the shortest sequence of words (i.e. one having the smallest possible number of words) which corresponds to a given number and a given list of words. The correspondence is described by the picture above.

Input

The first line of input file **PHONE.IN** contains the call number, the transcription of which you have to find. The number consists of at most 100 digits. The second line contains the total number of the words in the dictionary (maximum 50000). Each of the remaining lines contains one word, which consists of maximally 50 small letters of the English alphabet. The total size of the input file doesn't exceed 300 KB.

Output

The only line of output file **PHONE.OUT** contains the shortest sequence of words which has been found by your program. The words are separated by single spaces. If there is no solution to the input data, the line contains the text 'No solution.'. If there are more solutions having the minimum number of words, you can choose any single one of them.

Example

There is only one solution to the input file **PHONE.IN** containing

```
7325189087
5
it
your
reality
real
our
```

written in the output file **PHONE.OUT**, which is

```
reality our
```

(next possibility 'real it your' corresponding to the same number is longer).

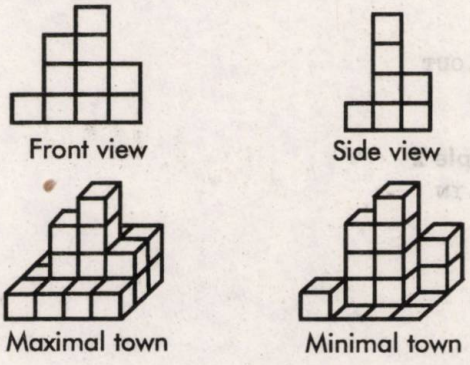
If the number is '4294967296', the only correct result is:

```
No solution.
```

because no given word contains the letters g and h which are necessary to obtain the digit 4

CEOI9905: Block Town

Children like playing with blocks (cube wooden bricks). They usually build high towers, but small Johny dreams of different plans. He is going to build a big town. His daddy has bought him a rectangular table; its width is K blocks and its length is L blocks exactly. Johny decided to project a plan of such a town before he starts building the town itself. He has drawn a square-shaped network on the table consisting of K×L squares. He wants to place the towers consisting of one or more blocks on some of the squares of the network drawn; the remaining squares will be empty. Because of the table being so large, Johny is not going to plan exactly for every square how many blocks he will put on it. He only wants to decide about front and right sight shapes of his town. He drew these two views (two-dimensional projections of the planned town) on a paper. You can see an example of these drawings and the adequate town made of wooden bricks in the pictures:



Johny's daddy is afraid that they don't have enough blocks to finish building Johny's planned town. You are asked to write a program to compute the minimal and maximal amount of blocks with which a town corresponding to Johny's plans can be built. Moreover, the program can decide about the possibility of building a town satisfying the views.

Input

The first line of input file **TOWN.IN** contains two positive integers K, L - the width and the length of the table (expressed as numbers of bricks). Neither the width nor the length of the table is greater than 100000 bricks. The following lines of the input file contain the description of the front view of the town. The description consists of a series of heights of visible buildings on each square from the left to the right (the height is measured by the number of the blocks, too). There is only one number on each line, i.e. the number of the lines with the front view description of the town equals K -- the width of the table. Similarly the next L lines of the input file contain the right sight view of the town. The heights of the wooden block towers are now specified from the front line to the back line. You may suppose there is no building in the town with height exceeding 5000 blocks. The maximal number of blocks needed for building the entire town does not exceed 2000000000.

Output

Output file **TOWN.OUT** contains only one line. If it is not possible to build a town with the front and right sight views given, only a text 'No solution.' is written there. In the other case two numbers will be written on the line and separated by a single space. The first one is the minimal and the second one is the maximal amount of blocks small Johny can use to build his town in accordance with his project.

Example 1

TOWN.IN

```
4 3
1
3
4
2
1
4
2
```

TOWN.OUT

```
10 21
```

Example 2

TOWN.IN

```
2 2
4
1
1
3
```

TOWN.OUT

```
No solution.
```

CEOI9906: Sightseeing trip

There is a travel agency in *Adelton* town on the *Zanzibar* island. It has decided to offer its clients, besides many other attractions, the possibility of sightseeing the town. To earn as much as possible from this attraction, the agency has accepted a shrewd decision: it is necessary to find the shortest route which begins and ends at the same place. Your task is to write a program which finds such a route.

In the town there are N crossing points numbered from 1 to N and M two-way roads numbered from 1 to M. Two crossing points can be connected by multiple roads, but no road connects a crossing point with itself. Each sightseeing route is a sequence of road numbers $y_1, \ldots, y_k$, $k>2$. The road y_i $(1 \le i \le k-1)$ connects crossing points x_i and x_{i+1}, the road y_k connects crossing points x_k and x_1. All the numbers $x_1, \ldots, x_k$ should be different. The length of the sightseeing route is the sum of the lengths of all roads on the sightseeing route, i.e. $L(y_1)+L(y_2)+\ldots+L(y_k)$ where $L(y_i)$ is the length of the road y_i $(1 \le i \le k)$. Your program has to find such a sightseeing route, the length of which is minimal, or to specify that it is not possible, because there is no sightseeing route in the town.

Input

The first line of input file **TRIP.IN** contains two positive integers: the number of crossing points $N \le 100$ and the number of roads $M \le 10000$. Each of the next M lines describes one road. It contains 3 positive integers: the number of its first crossing point, the number of the second one, and the length of the road (a positive integer less than 500).

Output

There is only one line in output file **TRIP.OUT**. It contains either a string 'No solution.' in case there isn't any sightseeing route, or it contains the numbers of all crossing points on the shortest sightseeing route in the order to pass them (i.e. the numbers x_1 to x_k from our definition of a sightseeing route), separated by single spaces. If there are multiple sightseeing routes of the minimal length, you can output any one of them.

Example 1

TRIP.IN

```
5 7
1 4 1
1 3 300
3 1 10
1 2 16
2 3 100
2 5 15
5 3 20
```

TRIP.OUT (one of the solutions)

```
1 3 5 2
```

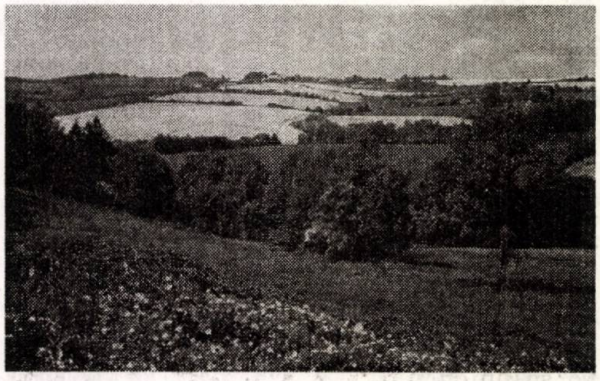
Example 2

TRIP.IN

```
4 3
1 2 10
1 3 20
1 4 30
```

TRIP.OUT (the only correct solution)

```
No solution.
```



Czech Highlands

BOI 2000

BOI 2000 took place in Ohrid, Macedonia and was attended by participants from Albania, Bulgaria, Cyprus, Georgia, Greece, Macedonia, Romania, Yugoslavia and The United States of America. Here you are presented the tasks the contestants had to solve during the two competition days.

BOI2K01: Market

Fisherman Clement has caught N fish and brought them to the fish market. The people from the fish market pack the fish in packages with different weights. Each package can have one or more fish.

Suppose you are the first customer and you can choose the weight of your package. How many possible different weights of packages exist with these N fish?

Limitations

- $N \leq 500$ is a positive integer representing the number of fish.
- $t_i \leq 1000$ is a positive integer representing the weight of the ith fish.

Time limit: 75 seconds

Input file: MARKET.IN

In the first line of the input file there is one integer number N. In each of the next N lines there is one integer number t_i.

Output file: MARKET.OUT

The output consists of one integer K representing the number of possible different weights.

Example

MARKET.IN	MARKET.OUT
5	27
800	
200	
354	
18	
182	

BOI2K02: Fractal Dimension

Clement learned at school how to compute the fractal dimension of a geometric figure. For this purpose he draws a unit grid with integer coordinates. Then he draws the figure on this grid with N vertices. These N vertices define a polygon that is not necessarily convex and whose lines do not intersect. He has to count the number K1 of unit grid squares that intersect with the figure. By *intersection* we mean that the square has a point in common with the interior of the figure. Then he deletes the grid lines with odd coordinates and counts the number K2 of the new grid squares that intersect with the figure.

The fractal dimension is computed by some formula depending on K1 and K2.

Clement wants to compute the fractal dimension of the *Ohrid Lake*.

Find K1 and K2.

Limitations

- $2 < N < 1000$ is a positive integer number representing the number of vertices.
- Each vertex is defined by two integer numbers x_i and y_i representing the *XY* coordinates ($-32000 \leq x_i, y_i \leq 32000$).
- Each vertex is connected with the one preceding it and the first vertex is connected with the last one by a straight line.

Time limit: 1 second

Input file: FRACTAL.IN

The first line of the input file has integer number N. In each of the next N lines there are two integer numbers x_i and y_i.

Output file: FRACTAL.OUT

The output consists of one line with two numbers K1 and K2.

Example

FRACTAL.IN	FRACTAL.OUT
4	110 30
0 0	
10 0	
20 10	
10 10	

BOI2K03: Diving

Each year there are diving courses in *Ohrid*. Different tasks are performed during the courses. One of them is to dive from one side of the rock through an underwater passage to the other side of the rock. The whole group of N students has only one oxygen bottle. No more then two persons can use the same oxygen bottle at same time. Each diver has its own speed of diving, so when two students dive together they dive at the speed of the slower one (one with longer diving time). There are M pairs of students that dislike each other and cannot dive together.

Find one schedule to move the group of students from one side of the rock to the other side in minimal time.

Limitations

- $N \le 6000$ is a positive integer and is the number of students in the group. Each student is denoted by $1, 2, \ldots, N$.
- $M \le 6000$ is a positive integer and is the number of student pairs that cannot dive together.
- t_i is a positive integer denoting the time required for the ith student to move from one side of the rock to the other side.
- The oxygen bottle can only be moved from one side of the rock to the other side by diving. The oxygen bottle has unlimited capacity. There is always at least one solution.

Time limit: 1 second

Input file: DIVING.IN

The first line of the input consists of two integers N and M. In each of the next N lines there is one integer t_i. In each of the next M lines there are two integers representing a student pair that cannot dive together.

Output file: DIVING.OUT

The first line of the output contains one integer, the minimal time required for moving all the students. The next lines of the output file are reflecting one possible schedule of moves with minimal time. Each line represents one move in the schedule. It consists of one or two integers denoting the student(s) who dive with the oxygen bottle from one side of the rock to the other side.

Example

DIVING.IN	DIVING.OUT
4 2	6
1	3 1
2	1
1	4 2
2	3
3 4	3 1
2 3	

BOI2K04: Bridges

Naum has returned to his village *Elshani* from abroad, where he has earned a lot of money. Everyday the people from *Elshani* go to work in one of the two neighboring villages *Peshtani* and *Lagadin*. There were a lot of paths in the mountain to their destinations. In order to help his neighbors and shorten their travelling *Naum* wants to build one bridge up to certain length Lmax.

Find where to build a bridge that satisfies the following criteria:

- There will be a new path from *Elshani* to *Peshtani* shorter than the existing shortest path
- There will be a new path from *Elshani* to *Lagadin* shorter than the existing shortest path
- The sum of the lengths of the new shortest paths will be minimal.

Limitations

- $3 \le N \le 2000$ is positive integer and is the number of points.
- The points are denoted by $1, 2, \ldots N$, where 1 is the starting point (the village *Elshani*), 2 is one destination point (the village *Peshtani*) and 3 is the other destination point (the village *Lagadin*). Two points can be connected by only one path. The length of the path is the shortest distance between the corresponding points. From each point there are at most 20 paths.
- $-32000 \le x_i, y_i \le 32000$ are integers representing the XY coordinates of the ith point.
- $M \le 5000$ is a positive integer and is the number of paths between the points.
- Lmax is a positive real number representing the maximum possible length of the bridge.
- Rounding errors will not produce irregular answers in test data.
- If there is a solution then it is unique.

Time limit: 15 seconds

Input file: BRIDGE.IN

In the first line of the input file there are two integer numbers N and M and one real number Lmax. In each of the next N lines there are two numbers x_i and y_i. In each of the next M lines there are two integer numbers representing the points connected by a path.

Output file: BRIDGE.OUT

If there is a solution, the output consists of two integers K_1 and K_2 representing the points where the bridge should be built.

If no bridge can be built with the proposed criteria then write 0 0 in the first line, and write two real numbers d_1 and d_2 in the second line as reals with no decimal places. d_1 is the shortest distance from point 1 to 2 and d_2 is the shortest distance from point 1 to 3.

BOI2K05: Spider

A spider and a bug are inside a glass box with dimensions L, M and N. They move on the surfaces of the box. They cannot fly and do not fall, no matter which surface they

are on. Find the strategy for the spider to catch the bug in minimal time.

Your program should play a game with the unit computer modules. The game is played in alternative moves. Your program simulates the movement of the spider, and the unit computer module responds with bug moves.

Limitations

- `L,M,N≤10000` are positive integers denoting the box dimensions in centimeters.
- The box is situated in the XYZ coordinate system with one node in the origin and three other nodes on separate coordinate axes.
- The spider and the bug can be positioned only on points with integer coordinates at the box surfaces as shown in the figure 1

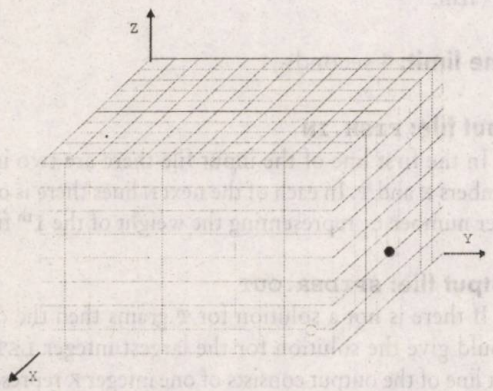


Figure 1: Box surfaces

- The spider can jump in one move as shown in the figure 2.

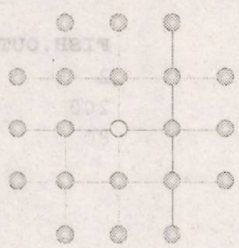


Figure 2: Spider moves

- The bug can jump in one move as shown in the figure 3.

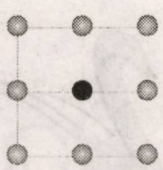


Figure 3: Bug moves

- Both the spider and the bug can move from one surface of the box to another one over the edge.
- The spider catches the bug if it reaches the same point where the bug is.

Time limit: 4 seconds

Input file: `SPIDER.IN`

The first line of the input consists of three integers `L`, `M` and `N`. The second line consists of three integers x_s, y_s, z_s representing the spider's starting position. The last line consists of three integers x_b, y_b, z_b representing the bug's starting position.

Output file: `SPIDER.OUT`

The call to the procedure `Finish` produces an output file. This file should not be modified by your program. The file will contain the number of moves, if the task is solved or the reason why your program failed.

Evaluation

The test run is not awarded with points if there are:

- Two consecutive calls of `movespider`.
- Two consecutive calls of `movebug`.
- Illegal spider moves.

Let `T` be the minimal number of spider moves required to catch the bug and let `S` be the number of spider moves produced by your program.

If `S<T+3` then the test run is awarded with 10 points. If `T+3≤S<1.2*T` then the test run is awarded with 8 points. If `1.2*T≤S<1.4*T` then the test run is awarded with 5 points.

Instructions for Pascal programmers

You will have a unit called `USpider.tpu`. At the beginning of your program you should set the following instruction:

```
uses USpider;
```

This unit contains 4 procedures: `Init`, `MoveSpider`, `Movebug` and `Finish`.

```
procedure Init;
```

The procedure `Init` should be called at the beginning of the program.

```
procedure MoveSpider(xns,yns,zns:Integer);
```

The procedure `MoveSpider` is used to set new coordinates of the spider move.

```
procedure MoveBug(var xnb,ynb,znb:Integer);
```

The procedure `MoveBug` implements the computer strategy and returns new coordinates of the bug move.

```
procedure Finish;
```

The procedure `Finish` should be called when the spider catches the bug. This procedure halts your program.

At the beginning your program must call procedure `Init`. Then your program should alternatively call

MoveSpider and MoveBug. When you call the procedure MoveSpider that sets the coordinates of the spider at the position of the bug then you call procedure Finish.

Instructions for C/C++ programmers

You will be provided with an object file USpider.obj and a header file USpider.h.

Before solving the task, you should open a new project. The project should be called Spider.prj and should contain two files: the USpider.obj and your code file. At the beginning of your code file, you should set the following instruction:

```
#include "USpider.h" or #include <USpider.h>
```

This .obj file contains 4 functions: Init(), movespider(), movebug() and Finish().

```
void Init();
```

The function Init() should be called at the beginning of the program.

```
void movespider(int xns,int yns,int zns);
```

The function movespider() is used to set new coordinates of the spider move.

```
void movebug(int *xnb,int *ynb,int *znb);
```

The function movebug() implements the computer strategy and returns new coordinates of the bug move.

```
void Finish();
```

The function Finish should be called when the spider catches the bug. This function halts your program.

At the beginning your program must call the function Init(). Then your program should alternatively call movespider() and movebug(). When you call the function movespider() that sets the coordinates of the spider at the position of the bug then you must call the function Finish().

Example

SPIDER.IN

```
10 12 8
10 11 1
5 12 2
```

Your Program interface with unit	*Bug coord returned*
`Init;`	
`movespider(9,12,2);`	
`movebug(xb, yb, zb) ;`	`(4, 12, 1)`
`movespider(7,12,1);`	
`movebug(xb, yb, zb);`	`(3, 12, 0)`
`movespider(5,12,0);`	
`movebug(xb, yb, zb);`	`(2, 11, 0)`
`movespider(3,11,0);`	
`movebug(xb, yb, zb);`	`(1, 10, 0)`
`movespider(1,10,0);`	
`Finish;`	

In this example the minimal possible number of moves is 5.

BOI2K06: Fish

Fisherman *Naum* has caught N fish. The weight of each fish t_i is measured at the coast. *Naum* works for one fishing company and he has right to take home at most T grams fish. He wishes to take as few as possible number of fish, but with maximal possible weight L(≤T).

Decide which fish to take home.

Limitations

- N≤500000 is positive integer and is the number of fish.
- T≤7000 is positive integer and is the maximum weight of fish he can take home.
- t_i≤35000 is a positive integer and is the weight of the ith fish.

Time limit: 4 seconds

Input file: FISH.IN

In the first line of the input file there are two integer numbers N and T. In each of the next N lines there is one integer number t_i representing the weight of the ith fish.

Output file: SPIDER.OUT

If there is not a solution for T grams then the output should give the solution for the largest integer L≤T. The first line of the output consists of one integer K representing the number of fish to be taken home. In each of the next K lines there should be one integer number k_i representing the weight of ith fish to be taken home.

Example

FISH.IN	**FISH.OUT**
10 280	2
300	200
10	80
30	
80	
200	
20	
20	
60	
10	
100	

8th BALKAN OLYMPIAD OF INFORMATICS
OHRID, MACEDONIA
ii.pmf.ukim.edu.mk/boi2000

Zadar, Croatia!

CEOI '98

We are now presenting to you the solutions to the tasks proposed for solving at CEOI '98. These are not the official solutions presented by the organizers; we are presenting to you the solutions proposed by our scientific coordinator, prof. dr. Horia Georgescu.

CEOI98-1: Squares

To each ith square we associate two variables of type record having the fields x and y, representing the coordinates of the upper-left and the bottom-right corner of the square, respectively. The problem of the visibility of the squares it reduced to the problem of the visibility of the segments P_iQ_i. It is obvious that $Q_i.x=P_i.x+lung_i$, and $Q_i.y=P_i.y-lung_i$, where $lung_i$ is the length of the square's side.

We also observe that the distance from the origin to a segment P_iQ_i is $(P_i.x+P_i.y)/\sqrt{2}$. The observation is based on the fact that the segment P_iQ_i is the diagonal of a square, so it always forms a 45 degree angle with the abscissa. Therefore in order to compare the distance of two segments with respect to the origin it is sufficient to take into account the quantities $P_i.x+P_i.y$.

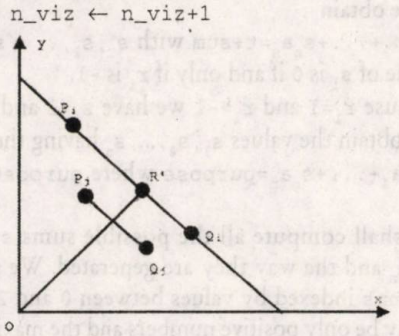
We will use a function Orientare which, called for two points *A* and *B* from the first [cadran], will return one of the values 1, 0, -1, depending on whether the slope of the line is higher, equal or lower than the slope of the line *OB*.

We begin by sorting the segments P_iQ_i by the slopes of OP_i in decreasing order.

We consider the segments P_iQ_i, i=1,...,n one by one and if such a segment is visible we increase by one the variable n_viz, which is initialized with the value 0.

Let P_iQ_i be such a segment. Let R' be the current point from P_iQ_i having the property that P_iR' is obturated by the segments P_jQ_j analyzed until this moment; initially R'=P_i. Because we are only interested in the slope of OR', instead of R' we will store points R which are updated with certain points Q_j. Then for the segment P_iQ_i we will perform the following computations:

```
R ← Pi
for j=1,n do
  if the distance from O to PiQi is smaller than that from O
     to PjQj
  then {the segment PiQi cannot be obturated by PjQj}
  else
    if the slope of OR is between the slopes of OQj and OPj,
       i.e. the line OR intersects the segment PjQj (see figure)
    then R ← Qj
    if the slope of OQi < the slope of OR
    then n_viz ← n_viz+1
```



We finally obtain the program source which is available at www.ginfo.ro)

CEOI98-2: Cards

Let q be the permutation of degree n read from the input and p the wanted permutation, i.e. the initial permutation which must be reconstructed.

It is essential that from the text of the problem we retain the:

- n is an odd number (1)
- the way in which the initial permutation p has been constructed. (2)

For example, (2) shows that the initial permutation p is not random, but has the property $p^n=e_n$, where e_n is the identical permutation of degree n.

The problem asks that, starting from the natural numbers n, s and from the permutation q, we compute that permutation p of degree n which, taken to the power 2^s, gives exactly q.

We observe that $(2^s,n)=1$ because n is odd. Then, based on a theorem taught in high-school algebra, there is a natural number u having the property: $2^su=1 \pmod n$.

Let t be the remainder of the division of 2^s by n; it is calculated as follows:

```
t ← 1;
for i=1,s do t ← (t+t) mod n;
```

Now u can be obtained as follows:

```
u ← 1;
while the remainder of the division of u*t by n is different
  from 1 do u ← u+1;
```

It follows that $2^s u = k*n+1$ for a natural number k. So we must compute the permutation p of degree n which taken to the power $2^s u$ is equal to q^u, i.e. $p=q^u$.

We obtain the program which is available at www.ginfo.ro)

CEOI9803: Substract

Let a be the vector having n elements and t be the value that must be obtained by making subtractions.

By subtracting some elements of a their sign may change. So, we are looking for the vector r having the property

$r_1a_1+r_2a_2+\ldots+r_na_n=t$ and $r_1, r_2, \ldots, r_n \in \{-1,1\}$ (1)

Let sum be the sum of the elements of the vector a: $a_1+a_2+\ldots+a_n=sum$. By adding the relation (1) and this relation we obtain

$s_1a_1+s_2a_2+\ldots+s_na_n=t+sum$ with $s_1, s_2, \ldots, s_n \in \{0,2\}$. The value of s_i is 0 if and only if r_i is -1.

Because $r_1=1$ and $r_2=-1$ we have $s_1=2$ and $s_2=0$. We have to obtain the values $s_3, s_4, \ldots, s_n$ having the property $s_3a_3+s_4a_4+\ldots+s_na_n=purpose$ where $purpose=t+sum-2a_1$.

We shall compute all the possible sums $s_3a_3+s_4a_4+\ldots+s_na_n$ and the way they are generated. We shall use a vector sums indexed by values between 0 and 20000 (the sums may be only positive numbers and the maximum sum is no more than 20000 because each of the maximum 100 terms has a value less or equal to 200). Initially we set sums[0]=-1, sums[i]=-2, for $1 \le i \le purpose$. When a certain value v is reached we set sums[v]=i where $v=v'+a_i$ and v' has been previously reached.

This way one may identify the elements a_i which have been added (actually the double of their value was added); this means one may determine the vector r satisfying the relation (1).

Now, using the vector r, we must determine the positions where the subtractions where made. For this, let us notice that if the equality $t=a_1-\ldots-a_i+a_{i+1}+\ldots+a_{i+k}-a_{i+k+1}-\ldots a_n$ holds, then making k subtractions on the position i we obtain $t=a_1-\ldots-b_i+a_{i+k+1}-\ldots a_n$ where $b_i=(\ldots((a_i-a_{i+1})-a_{i+2})-\ldots-a_{i+k})$.

Using these results we may describe an algorithm for solving the task: let pos be the current position where we make operations; at the beginning the value of pos is n and the following operations are repeated until pos becomes 1:

- we skip all the positions corresponding to values rpos = -1 by decreasing the value of pos
- we skip all the positions corresponding to values rpos = 1 by decreasing the value of pos; let no be the number of the positions skipped at this step
- we introduce in an array subtr, no times the current value of pos.

The subtr array must contain n-1 elements because n - 1 subtractions are to be made. At this moment t may be expressed as a "sum" in which only the first term has the corresponding r value equal to 1, so any further subtractions must be made on the first position. In conclusion we shall fill the rest of the array with the value 1.

The source code for this solution is available at www.ginfo.ro.

CEOI9804: Soldiers

Step 1. It is obvious that all the points (soldiers) must arrive on a horizontal line y=yf. It follows that we shall look for yf such that the sum of the distances from all the points to that line $\sum_{i=1}^{n} |yf-y_i|$ is minimum.

We shall decreasingly sort the array y, so we shall have $y_1 \le y_2 \le \ldots \le y_n$. Consider first only y_1 and y_n; it is clear that the sum of the distances to y_f is minimum if we choose as value of y_f any number in the interval $[y_1, y_n]$; the sum would be y_n-y_1, no matter which yf in the interval we choose. We continue with the pair y_2, y_{n-1}; as in the previous case the sum of the distances to y=yf is minimum $(y_{n-1}-y_2)$ if we choose for yf a value in the interval $[y_2, y_{n-1}]$. We continue in the same manner; if, in the end, we remain with a single number (n is an odd number) then the value of yf is the y coordinate of the corresponding point; if we remain with two points (n is an even number) we can choose as yf any of these numbers.

Hence yf may be chosen as the y coordinate of the $[(n+1)/2]^{th}$ point. Actually, we are not interested in finding the value of yf; we are only interested in the minimum value of the sum which may be obtained as follows:

```
sy ← 0
p ← 1
u ← n
while p < u do
  sy ← sy + yu - yp
  p ← p + 1
  u ← u - 1
```

Step 2. We reduced the plan problem to a line problem (on the line y=yf). The points having the x coordinates $x_1, x_2, \ldots, x_n$ must be moved on consecutive positions $xf+1, xf+2, \ldots, xf+n$. It follows that we shall look for xf such that $\sum_{i=1}^{n} |xf+i-x_i|$ is minimum, that is $\sum_{i=1}^{n} |xf-(x_i-i)|$ is minimum.

Repeating the same reasoning as in the plane problem, we shall store the values of the x array into another vector called xinit. Then we shall replace each value xi by the value x_i-i and by sorting this vector we shall obtain the value xf. The effort needed for bringing the soldiers from the positions $x_1, x_2, \ldots, x_n$ to the positions $xf+1, xf+2, \ldots, xf+n$ may be computed immediately as the sum of the values $|xf-(xinit_i - 1)|$.

The source is available at www.ginfo.ro.

CEOI9805: Roads

We shall define the tasks in terms of the graph theory.

Given a directed multigraph having the vertices 1, 2, . . . , n, every edge of the graph is associated a length and a tax that must be paid for passing through that edge and a sum k, find the shortest path from 1 to n (if it exists) so that the total tax be less or equal to k.

An edge is stored as a record called edge having the fields source, dest, length and cost having obvious significance.

The number of edges is nm and they will be stored in an array M. We shall use an array L (as a list) whose elements are records having the fields node (the current vertex where we arrived starting from 1), cost (the tax paid for arriving to node) and length (the length of the path from 1 to node).

We start with L_0 which has the value (∞, ∞, ∞); the purpose of this element is to ease the insertion of an element in the list. The head variable is used for keeping track of the position immediately after the last element in the list (the first which shall be eliminated); so initially we have head=1.

We shall use the array distmin for storing the current minimum distances (with taxes less than k) between the node 1 and each of the other nodes.

The elements of L are always sorted by the cost field; L(head-1) will have the lowest cost.

When adding a triplet (cost,node,length) we must check whether there already exists an element having the same node value but smaller value for cost and length.

After reading the edges, the vector M will be sorted by the source field and for each node i in the multigraph we identify the position start[i] where the first edge having the source i appears in the vector M.

The idea for solving this task is a breadth-first search of the multigraph, starting with a list L having only the element (0,1,0).

We pop the first element of the list (position head-1); let it be (cost,node,length). If length>distmin[node] then this vertex will not be expended. Otherwise we shall determine all the vertices j that can be reached directly from node (there exists an edge between node and j). We shall add to the list the triplets (cost',j,dist') where cost'=cost + the cost of the edge (node, j) and dist' =dist+ the length of the edge (node, j) only if cost'≤k and there is not any element (cost",j,dist") such that cost"<cost' and length"<length'.

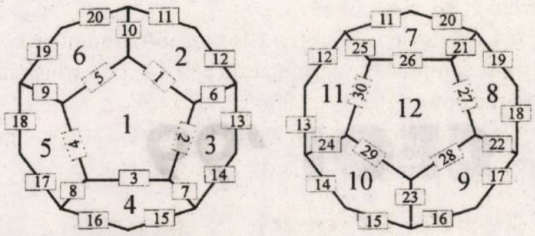
The algorithm ends when the list becomes void; the result will be distmin[n].

The source is available at www.ginfo.ro.

CEOI9806: Ball

We can consider this task as a generalization of the map coloring problem because the purpose is to color the edges using the colors on the edges of the tiles. This is why we choose to apply the backtracking method (the recursive variant).

We start by numbering the edges as shown in the following figures:



We shall store the edges of each side in a vector called mvec. We shall first place the tile from side 1, then the tile from side 2 and so on until we reach side 12. In the matrix mnew we shall store the edges that are colored when we place a tile on a side. The vector x is used for storing the colors corresponding to the edges (if an edge is not colored yet, the corresponding value is 3).

The tiles will be stored in the array tiles having five fields which contain the colors of the edges of each tile.

The boolean vector used stores the current status of the tiles (they have been used or not).

The procedure Back, called from the main program by Back(1) is described in what follows:

```
procedure Back(k)
 if k = 13 then write_result; stop
 else
  for i=1, 12 do
    {try to place the tile i on the side k}
    if used(i) then
      p ← tiles(i)
      for j=1, 5 do
        {the five positions of the tile}
        if cont(k, p) { the condition for
                        continuing }
        then
        * coloring the new reached edges
          used(i) ← true
          back(k+1)
        * decoloring the edges just colored
          used(i) ← false
          {rotate the tile p}
```

The condition for continuing (cont(k,p)) checks whether placing the tile p on the side k respects the coloring of the already colored edges.

Remark

The way we count the edges reduces the number of computations. One may notice that the characteristic vectors from mnew have the value 1 only on the first positions. Hence instead of using mnew it is sufficient to store only the number of 1's we have (the number of edges will be colored when placing a tile on the current side).

The source file for solving this task is available at www.ginfo.ro.

Brno, Czech Republic!

CEOI '99

We also present you the solutions of the tasks proposed for solving at CEOI '99. These are not the official solutions presented by the organizers; we present you the solutions proposed by one of the Romanian participants, Dumitru Bogdan. He obtained a silver medal at CEOI '99.

CEOI9901: A Game on the Chessboard

After reading the task description for the first time, the most suitable solving method seems to be *Branch & Bound*. There are several disadvantages if we use this method: it is relatively hard to implement, a lot of memory is needed and perhaps the program would not run in time.

Estimating the total number of possible configurations, one may notice that this number is equal to the number of 8-combination over 16-set, i.e. `12870`.

One may also notice that every configuration may be coded with a number representing the decimal equivalent of the binary number obtained by liniarizing the configuration. The idea of the solving algorithm is an "expansion" based on the idea of the *Lee* algorithm. We shall use a queue initialized with the initial status of the board. At each step we shall add all the configurations that may be obtained from the current configuration if they have not been obtained before. This way the configuration is obtained in a minimum number of steps. The process ends when the final configuration is obtained (it is obvious that there always exists a solution - moreover it can be easily proved that the solution is obtained after `24` steps or less).

For a better understanding of the solution one may consider the `12870` configurations as the nodes of a graph. We consider an edge between two nodes if the corresponding configurations may be obtained one from the other in a single step. The algorithm implements a breadth-first search of the graph starting from the initial position and ending on the level that contains the final position.

The procedure `Load` reads the initial and the final positions of the board.

The procedure `Equal` treats the trivial case when the two positions are identical; in this case the number `0` is written to the output file and the program is halted.

The explanations of the procedures `Board_to_Number` and `Number_to_Board` are a bit delicate. We considered all the 12870 possible configurations decreasingly sorted by the decimal equivalent of the binary number obtained by liniarizing such a configuration and we associated each configuration its index. The procedure `Board_to_Number` has a configuration as parameter and returns its index in the sorted array of configurations; the other procedure, `Number_to_Board`, has a number as parameter (from `1` to `12870`) and returns the corresponding configuration. Both procedures use the matrix `C` computed using the procedure `Comp_Comb`; `C[i,j]` has the value *j*-combination of a *i*-set; the matrix is determined using *Pascal*'s triangle.

The procedure `Init` initializes the queue `V` with the initial configuration. We shall also use a characteristic vector `S` which indicates whether a certain configuration has been obtained or not, as well as a vector `T` which indicates for each configuration the previous one (the one from which it was obtained).

The procedure `Work` fills the queue until the solution is found. It uses the procedure `Move` which, having a configuration as parameter, computes all the configurations that may be obtained in one step from that configuration.

After the solution is computed the procedure `Build` is called; this procedure builds the path from the initial status to the final status. The solution is written to the output file using the procedure `Write_solution`.

Every configuration may be obtained from at most other `24` configurations (the number of neighbor cells interchanges in a square matrix having the dimension 4). Hence the complexity of the algorithm is *O(Total number of configurations)*. This number is a constant (12.870), so the complexity is *O(1)*. The program runs almost instantly for any test.

The source code for solving this task is available at www.ginfo.ro.

CEOI9902: Orders

This task, even if it had a description difficult to understand, was very easy to solve. The specification "all possibilities" indicates that the problem should be solved using the backtracking method. All that had to be done was opti-

mizing the program as much as possible because the number of possible solutions was very large.

We define a procedure `Load` which reads the available characters, and stores in the vector `nr` how many characters of each type are. The procedure `Preprocess` builds the vectors `chars` and `how_many`; the vector `chars` is used for storing the distinct characters in ascending order (their number is stored in the variable `l`); the vector `how_many` retains how many characters of each type we have; the total number of characters is stored in the variable `s`.

The recursive procedure `Rec` implements the backtracking method. If we are at the level `s+1` we have a solution which is written to the output file. Otherwise we fill the current level (denoted by `k`) using all the characters available at that time; the order of the characters is given by the sorted vector `chars`. Then we decrease the corresponding element in the `how_many` vector and recursively recall the procedure `Rec`. The ascending order of choosing the characters guarantees the lexicographic order of the solutions.

The procedure `Bkt` opens the output file for writing and associates a buffer (needed because of the size of the file - `2 MB` maximum); then it calls the procedure `Rec`.

Every solution is written by the procedure `write_solution` which is called every time the finishing condition of the recursive procedure is validated.

The complexity of the solution: for each character added to a solution we may test all the distinct characters (maximum 26 - constant denoted by `nmax`); the number of operations performed is proportional with *O(Output * nmax)* where *Output* denotes the size of the output file. This complexity assures passing all the tests used for the evaluation.

The source code is available at www.ginfo.ro.

CEOI9903: Parity game

Most contestants considered this task as the "smartest" one proposed for solving during the contest.

After reading the task description, the first idea for solving it was to merge the intervals for which the upper bound of one of them and the lower bound of another are consecutive numbers; this way we try to obtain a contradiction (an interval which we know that contains an odd number of `1`'s and an even number of `1`'s - this fact is obviously impossible).

The biggest disadvantage of this method is that it creates new intervals all the time and there is not enough memory to store all these intervals.

One may notice that, instead of merging intervals to obtain new larger ones, we may "subtract" intervals thus obtaining smaller ones; it is easier to work with smaller intervals. For simplicity we shall say that an interval is even if it contains an even number of 1's and odd otherwise. The basic principle of the solution is: having the parities of the intervals `[x,y]` and `[x,z]` (`y<z`) we retain only the parities of the intervals `[x,y]` and `[y+1,z]` (the interval which has the minimum length and the "difference" of the two intervals). It is obvious that every interval manipulation can only be based on coincidences of their bounds (no contradiction may appear in another place).

Now we must see how can the intervals be "reduced" efficiently.

The best solution seems to be the following: we retain a vector `a` having as elements intervals (given by their bound and parity). This vector is always sorted by the lower bound of the intervals. When reading a new interval, it is added to the end of the vector.

Then we find an interval (if there is one) which has the same lower bound as the just read interval. Denoting by `I` this interval and by `J` the new one, three cases appear:

1) the upper bound of the interval `I` is greater than the upper bound of the interval `J`; in this case the intervals are interchanged and the algorithm should check the third case
2) the upper bound of the interval `I` is equal to the upper bound of the interval `J`; here we have two subcases:
   2.1) if the two intervals have the same parity the new information is redundant and is eliminated
   2.2) otherwise we found a contradiction; the number of read answers (`l`) is written and the program is halted
3) the upper bound of the interval `I` is lower than the upper bound of the interval `J`; in this case the interval `J` is replaced by an interval having the properties:
   - the lower bound is one plus the upper bound of the interval `I`
   - the upper bound is equal to the upper bound of the interval `J`
   - the new interval is even if `I` and `J` have the same parity and odd otherwise.

The scan of the intervals vector continues reducing the intervals where this is possible. After we finish reading the intervals, no contradiction was detected; the number of read answers is written and the program is halted.

The procedure `Load` loads the intervals one by one, each interval being manipulated by the procedure `Work`. This procedure adds the new interval to the position `l` of the vector after incrementing the variable. We scan (going backwards) the interval vector `A` until we find the first interval whose lower bound is at most equal to the lower bound of the last read interval. Depending on the relationship between the upper bound of this interval and the upper bound of the last interval, we have three cases that are treated as described previously. Then, we consider the next interval and make all the possible reductions.

When detecting a contradiction, the procedure `Stop` is called; this procedure writes the number of read answers and halts the execution of the program.

The complexity of this algorithm is $O(n^2)$; but, for a certain test, it is improbable to scan the entire vector at each step (this is the case when the intervals are given decreasingly sorted by their lower bounds). Anyway, a program which implements this algorithm should use only a fraction of the allowed running time for the tests used for evaluation.

The source is available at www.ginfo.ro.

CEOI9904: Phone numbers

This task was proposed for testing the dynamic programming knowledge of the participants, as well as the capability of manipulating large data structures.

The maximum `50000` words, each having at most `50` characters, can not be stored in a vector of pointers to strings because we would need `50000*50=2500000` bytes (more then `2.4 MB` of memory). The lists are not a solution either, because a "link" between two cells needs `4` bytes of memory. The only solution for storing the data was a character matrix; the words where written one after the other in this matrix. This solution is based on the fact that the total length of the words is no more than `300000` characters. A `50000` elements vector is also needed for storing the lengths of the word.

Apparently this method has many disadvantages: a word can only be accessed after computing its offset from the first cell in the matrix and a word may need more than one line in the matrix. Actually the following algorithm will efficiently use this matrix.

The solving method is described in what follows: for each position in the phone number we retain the minimum number of words needed for coding the number until the respective position and the last word used. We shift the phone number and, from the current position, we try to optimize using all the words we have by shifting the created matrix. The algorithm is a variant of the dynamic programming known as the backwards method. If we reach the last position we have a solution; otherwise there is no solution.

The procedure `Load` reads the given words and stores them in the matrix. We have chosen a matrix having `30` lines, each having `10000` characters. This way a word (which has a maximum length of `50` characters) will be stored using at most two lines in the matrix.

The procedure `Init_OK` initializes the matrix `ok` used for testing whether it is possible to code a certain digit using a certain character.

The procedure `Init` initializes the vector `v` (`v[i]` stores the minimum number of words used for coding the subsequence formed using the first `i` digits of the phone number).

Having the optimal solution for the first `i` digits, the procedure `Work` tries to add all the available words to the solution. These words are considered in the order given by the matrix and are checked using the matrix `ok`. If a certain word matches an update is made.

The procedure `Impossible` determines whether the problem has no solution; in this case a corresponding message is written and the program is halted.

The procedure `Reconst` finds the solution; this is not an easy task at all. First we determine the order numbers of the used words (they are stored in the vector `use`) and then sorts this word (the sorted vector is `cuse`).

In the procedure `Write_solution` we shift the matrix which stores the words; when a used word is found it is written to the output file.

For the maximum `100` digits of the phone number we shift the matrix of words whose dimension is at most `300000`. The complexity of the algorithm is *O(Number of Digits * Input)*; practically, the optimizations made (we stop the cycle at the first missed match) reduce the running time. The program which implements this algorithms runs almost instantly for all the tests used for evaluation.

The source is available at www.ginfo.ro.

CEOI9905: Block Town

This was one of the most beautiful tasks proposed for solving at *CEOI '99*; the type of this task is known as "idea task"; for solving this task one needs a lot of imagination, 3D view and large data structures manipulation. The program is very short but the reasoning on which it is based is very interesting.

First we must establish whether we have a solution or not. For this, consider an arbitrary town plan; it is obvious that it will have at least a tower having maximum height. This tower will be the highest tower from the front view and from the right view. Hence a necessary condition for the existence of the solution is that the two given array have the same maximum. In what follows we shall prove that this condition is also sufficient.

Considering this condition as being satisfied we shall build a town having a minimum number of blocks. Considering the right view, we denote by `l` the line of the highest tower (if there are two or more such lines, we choose one of them). Considering the front view, we denote by `c` the column of the highest tower (if there are two or more such columns we choose one of them). For satisfying the front view conditions we put on the line `l` a tower having the read height. The front view conditions are satisfied using a minimum number of blocks (the sum of the elements of the first read array).

We now consider the right view; the condition for the line `l` is satisfied because the tower having maximum height may be seen. We scan the rest of the lines; suppose that for a certain line a tower having the height `h` is needed. We look for the height `h` in the front view conditions. If we find such a height we denote by `cl` the corresponding column and the tower will be translated to the considered line. This way, the right view condition is satisfied; for the front

view nothing has change; the tower has been moved but it remained in the same column. If we do not find the height h then we must build a tower having the height h. We place the tower on the column c of the considered line. From the front view, the new tower is not visible because of the highest tower. In this second case, the minimum number of blocks is incremented by h.

Now, let us build the town having a maximum number of blocks. Again, we shall first satisfy the conditions of the front view. We shall scan the columns and we shall place on every line a tower having the given height. We compute the number of used blocks as the sum of the elements of the first read array multiplied by the number of lines of the matrix. From the right view we see, on each line, a tower having maximum height. We scan the lines and, for each line, we "cut" every tower whose height is higher then the given height for that column.

In the program we used a small implementation trick: the given arrays are not stored as they are read; we use two vectors a and b; we denote by a[i] the number of towers having the height i at the front view and by b[j] the number of towers having the height j at the right view. This is possible because the order of lines or columns is not important. It is also necessary because the matrix may have the dimension 100000*100000, but the maximum height is only 5000.

The procedure Load reads the heights from the two views and builds the vectors a and b.

The procedure Impossible checks whether we have a solution or not; if there is no solution a corresponding message is written to the output file and the program is halted.

The procedures Minimum and Maximum compute the requested values denoted by min and max.

The results are written to the output file by the procedure Write_solution.

Most of the running time is consumed reading the data; the rest of the operations performed are proportional to the maximum height which is at most 5000. Hence, the complexity order of the algorithm is *O(Input + hmax)*. The program runs in time for any test.

The source is available at www.ginfo.ro.

CEOI9906: Sightseeing trip

This is a classical graph problem: computing the minimum weight cycle in a graph.One may notice that, if there is more then one edge between two nodes, we shall never use two of them when computing the requested cycle (if we use two such edges then the cycle will consist only in these two nodes because it must by a simple cycle; this fact contradicts the restriction k>2 in the task description - the cycle must have at least three nodes). Hence, if there is more then one edge between two nodes, we shall consider only the one having the minimum weight.

First we shall compute the minimum weight cycle containing the node 1, then we shall compute the minimum weight cycle containing the node 2 and so on until we compute the minimum weight cycle containing the node n. From these n cycles we shall choose the one which has minimum weight.

How can we compute the minimum weight cycle containing the node i? Considering an arbitrary node j contained in this cycle we may consider the cycle as the union of two paths from i to j, paths having no other common nodes but i and j. It is obvious that one of these two paths is the shortest path from i to j (otherwise it could be substituted by the shortest path and the weight of the cycle would be smaller). The other path is the second shortest path which has a weight at least equal to the one of the first path. Considering this second path as the reunion of paths (let k be the number of these paths) it will consist in k-1 optimal paths and a path having a weight at least equal to the corresponding optimal path. The idea is to use the Dijkstra algorithm having the source i. This algorithm computes the shortest paths between i and all the other nodes in the graph. The reunion of these paths is a tree having the node i as root (this fact is obvious - the parent links computed by this algorithm generate a tree). We shall have m-(n-1)=m-n+1 edges which are not part of the tree (m is the total number of edges). Consider any such edge and the cycle consisting in this edge and the shortest paths from the edge nodes to the root of the tree. If the weight of such a cycle is smaller then the weight of the current minimum cycle, then the minimum cycle is updated.

The procedure Load reads the input file and stores in the matrix A the minimum weight edge between each pair of nodes.

The procedure Dijkstra implements the Dijkstra algorithm having a given source node.

The procedure Update computes the weight of each cycle consisting in an edge not included in the tree and the shortest paths from the edge nodes and the root of the tree.

The main procedure is Work; it calls the procedure Dijkstra n times considering each node as a source.

If the graph consisting only in the minimum weight edges between each pair of nodes is a tree then there is no solution; this case is detected by the procedure Impossible. Otherwise we have a solution which is computed by the procedure Cycle; the solution is written by calling the procedure Write_solution.

The Dijkstra algorithm is used n times; the complexity of the Dijkstra algorithm is $O(n^2)$. The number of the edges outside the computed tree is also proportional to n^2. Hence, the complexity of the algorithm is $O(n^3)$; the maximum value of n is 100, so the algorithm runs in time for any test.

The source is available at www.ginfo.ro.

Talking to Mr. Horia Georgescu...

PAST & FUTURE

Mr. Horia Georgescu is professor at the Academy of Economic Studies in Bucharest and at the Ovidius University in Constanța. He is also director of the Computer Science College of the Bucharest University and scientific coordinator of the ACM team from the Bucharest University. We asked him several questions about the past and the future of the CEOI.

It is not a secret that you are one of the initiators of the Central European Olympiad in Informatics. Before the appearance of CEOI, there were already two international programming contests (IOI and BOI) attended by Romanian teams. Please tell us why did you consider that this third contest was necessary?

I was the leader of the Romanian-IOI team from 1990 to 1994. You have to remember that before 1990 people from the former communist countries did not have the opportunity to travel or even communicate with people living abroad. As a consequence, the students of our team were very shy and even afraid of all they saw in the foreign countries where the IOI took place. So, I firstly proposed to organize BOI and afterwards, together with Mrs. Clara Ionescu, the CEOI. This way, the students and their teachers had the opportunity to learn more about other countries, about their culture and history and make new friends. In the same time they got accustomed to the kind of problems submitted at different national contests. Last, but not least, it was a very good training for IOI.

The first CEOI took place in Cluj, in 1994. Do you have special memories from these times?

Yes, I mainly appreciated the efforts made by the Computer Science Highschool in Cluj, by all the staff of this highschool and especially by Ms. Clara Ionescu, to organize such a contest for the first time. I think it was successful, and all the participants agreed with it. It was a big challenge, because the failure of the first contest would have implied the failure of CEOI on the whole.

We know that you participated in the CEOI'96 in Bratislava. The members of the Scientific and Evaluation Committees were all university students. What did you think of it at that time?

Of course I thought it was a good idea. Young people have to be assigned responsibilities too, in order to prepare them for the future. Even more, the dialogue with the participants was more direct. At the same time, the students have to be supervised and this happened in Bratislava, even if the supervisors did not seem to exist.

CEOI 2000 will also be managed mostly by students. Do you have confidence in their ability to do a good job?

Yes, because they are very well prepared and proved this by outstanding results in previous national and international contests. They already have some experience, they have had the same role in our national contests in the last few years.

We know that you no longer participate in the training camps of the Romanian high-school students, but you train university students for the ACM contest. Are there any important good things that are used at that contest and could be used at the CEOI, too?

There are a lot of differences between highschool students' programming contests and ACM. The most important is that at ACM there are not individuals, but teams. However, each team has only one computer, so the members of the team have to distribute tasks among themselves and in the same time to cooperate well.

What changes do you think will appear in the future of the high-school students' programming Olympiads?

This subject has to be discussed in Cluj. I will propose that for the future, CEOI should have teams and not individuals. Teamwork is important for highschool students. That is the way they will work in the future and therefore they have to be prepared for such kind of work. Having the computer as the only friend makes people selfish.

As scientific coordinator of the "GInfo" magazine, you insisted that it should include the problems submitted at different national and international contests. In which way do you think this fact will help Romanian students to prepare themselves for future contests?

In one way, it is obvious that by doing so, the Romanian highschool students have the opportunity to exercise their abilities at different kinds of problems; the same is true for their school teachers. They have to know that software development is and will be important in the future and this does not mean only writing applets or producing friendly interfaces; besides these, a lot of problems are to be solved. On the other hand, in GInfo the stress is not only put on this subject, so the journal is attractive for a large number of highschool students.

About CEOI in CLUJ-NAPOCA

Ms. Anca Voicu is the director of the Department of Information Technology and Networking in the Romanian Ministry of National Education. Until April 2000 she was the computer science inspector and was involved in organizing many national events similar to the CEOI. We shall now see what her opinions about the 7th edition of the CEOI.

As a representative of the Ministry of Education, you are the main organizer of the CEOI 2000 which will take place in Cluj. Taking into account the fact that in 1994 the first edition of CEOI took place in Cluj too, why do you think this city has been chosen again?

It is a difficult question, and e the answer requires a short explanation. Initially, the place chosen for CEOI 2000 was the *Felix Baths* touristic resort. It is a resort much closer to the western border of the country and the organizational elements (accomodation, meals, contest, recreational moments) would have taken place in a very pleasant place of high touristic attraction. Unfortunately, we encountered numerous organizational obstacles, so in *January 2000* we were forced to look for another place. It was very difficult to find a better alternative because an essential requirement was the availability of several people who had to know what the organization of an international contest meant and who could also assume the responsability to organize the competition in such a short time, starting practically from zero. This is why we have chosen the city of *Cluj*, the place where this international contest was born and where *Mrs. Clara Ionescu*, one of the persons who initiated this contest, engaged herself again to lead this activity to the end in the best conditions.

We assume that organizing such a contest is not among the easiest tasks to be undertaken. Who supported you from the Ministry of Education? What about other organizations?

These kinds of actions are difficult to organize because we don't have all the money required to organize it perfectly. This is why *Computer Press Agora* joined us as a main collaborator, having the difficult task of informing the larger audiences about this competition and to attract as many sponsors as possible. At the same time, the *Cluj County School Inspectorate* expressed its availability to support us in our undertaking.

For almost five years you have been a computer science inspector in the Ministry of Education, responsible for the teaching of computer science in Romanian high-schools. What is, in your opinion, the "key to success", that is, the key to the good results obtained by the Romanian contestants at the internatinal informatics contests?

The Romanians are an intelligent nation, aren't they? Many others have said this before me. But the success of the Romanian contestants is the result of their hard work. This work is also done by the teachers who prepare them.

Unfortunately, many of the former olympics, who have obtained numerous awards at international, contests choose to continue their studies abroad. What are their motives? Do you think that we could do something to keep them home?

This subject is always discussed. Didn't Romanians go abroad before? Of course they did, but nowadays, because of our conquered liberty, many more choose to leave. I don't think it's important how many leave but why they leave. First, many leave to develop professionally, others leave for a better salary, others just because it is "fashionable". It would be very important to find out how many of those who left come back, and if they don't come back, how many of them work in projects for the Romanian state.

I have said on other occasions that their leaving proves that the Romanian education system is good and very competitive. It is very true that we should find solutions to keep them home, but these solutions cannot be restrictive measures but excellent work offers. Maybe in time, a balance will be found...

Programming CONTEST

Last year GInfo organized the first edition of a programming contest for high-school and university students. The competitors were all from Romania, but this year contestants from all over the world may participate. For further details read the following pages or visit our web site at www.ginfo.ro.

General information

The *Agora Programming Contest* (*APC*) is an international competition for high-school and university students. It is organized by the *GInfo Editorial Office* and the *Computer Press Agora Publishing House*.

There will be six rounds, at each round the contestants will have to solve three tasks and submit their solutions by e-mail; the e-mail address of the *APC* is contest@ginfo.ro. After the six e-mail rounds there will be a real-time round held at the *Computer Science High-School* in *Cluj-Napoca*, *Romania*; only the top ten ranked contestants will participate in this round. The competition tasks are proposed by the *GInfo Editorial Staff*, in collaboration with the students and teachers involved in the training of the Romanian students who participate in various national and international programming contests.

Registering

For participating in the *Agora Programming Contest* you must first register. For registering simply send an e-mail having the following structure:

```
From: your name <e-mail>
To: contest@ginfo.ro
Subject: subscribe APC
```

and in the message body:

```
First name: your first name
Last name: your last name
Gender: M/F
E-mail: your e-mail address
City: your city
Country: your country
```

We shall send you an e-mail containing your registration code (ID); this code will be used for your identification.

If there is a problem with the APC e-mail address, you may also try one of the following addresses:

- office@ginfo.ro
- clara@cs.ubbcluj.ro
- skortzy@email.ro
- smihai@asterix.obs.utcluj.ro

Regulations

The accepted programming languages at the *APC* are *Pascal* and *C/C++*. Your source files will be compiled using *Borland Pascal 7.0* or *Borland C++ 3.1*.

The contestants from outside Romania will receive the task descriptions by e-mail on the first day of every month (*October 1st 2000 - March 1st 2001*) and will have to submit their solutions until the last day of the respective month. The contestants will be notified of any changes concerning these dates.

Only one solution will be accepted for each task. If a contestant will send more than one solution for a certain task, only the first one is considered and the rest of them are ignored.

The *GInfo Editorial Staff* reserves the right to disqualify the contestants that submit identical (or almost identical) solutions.

Your source files must not contain any assembler code; if a source file contains such code then the contestant will receive 0 points for the corresponding task.

The programs generated using the contestants' source files are not allowed to modify the timer. Any contestant who tries to modify the system timer will be disqualified.

If you have questions regarding the task descriptions you may submit them using the *APC* e-mail address.

You are only allowed to ask "YES/NO" type questions.

If an answer to a question may be found in the task description then the answer you will receive is

NO COMMENT.

If the *GInfo Editorial Staff* considers that an answer to a certain question concerns all participants then the answer will be GENERAL ANOUNCEMENT and will be followed by an e-mail which will clarify the question. This e-mail will be received by all the participants.

Submitting solutions

For submitting the solutions of a certain round you must send an e-mail to the *APC* address having as *Subject* your registration code followed by the text `Round X`, where `X` will indicate the number of the round (`1-6`). The e-mail must have attached a `.RAR` or `.ZIP` archive file having the name `yyyyRx.ext` where `yyyy` is your registration code, `x` is the round number and `ext` is the extension (`RAR` or `ZIP`). The archive must contain a directory called `yyyy` (your registration code). This directory must contain four files: three source files corresponding to the three tasks of the round and one `.BAT` file which contains the command lines you want us to use for compiling your source files. Normally the `.BAT` file will contain three lines. The names of the source files are given in the task descriptions and the name of the fourth file is `ROUNDX.BAT`. For security reasons self-extracting archives are not accepted.

The archive file must not contain any other files or directories, so you cannot submit any auxiliary files with your source codes. If your program needs to create auxiliary files then these files must have the extension `.TMP` and must be deleted by your program. If this rule is broken the contestant will receive 0 points for the corresponding task even if the solution is correct. If the *GInfo Editor Staff* considers that a contestant endangers (by overwriting certain files, creating very large files etc.) the competition progress then the contestant may be disqualified.

All the submitted files become the property of *Computer Press Agora* and can not be used for commercial purposes without a written permission from the publishing house.

Evaluation

For each task you may obtain at most `100` points; the maximum total points for a round is `300` and `1800` for the entire contest. For testing each task we shall use between `5` and `20` test data sets; a certain test is considered to be passed if your program generates a correct output for the input data corresponding to that test. If a test is passed then a certain number of points are added to the total points of the corresponding task.

The computer used for evaluation has the following configuration: *Intel Pentium™ III 450 MHz* Processor, `128 MB RAM`, `13 GB HDD`. It is guaranteed that during the evaluation process at least `300 KB` of conventional memory and `100 MB` of disk space are free. The operating system we use on this computer is *Microsoft Windows Me* (*Millennium Edition*). If a program performs an illegal operation then the contestant will receive `0` points for the corresponding test data set.

Advice

Pascal programmers should never use the `CRT` unit because their programs will stop returning the run-time error: `Error 200: Division by zero`. This is caused by a well-known *Borland Pascal 7.0* bug; no task proposed for solving in our contest needs to use this unit.

When using the several data types (such as `Comp`), the `8087` mode must be active. The environment options allow setting this mode as implicit mode, but it is not the default mode. Pascal programmers are asked to include in their source code the compiler directive `{$N+}` when using this mode. Otherwise when compiling the source code, the `Error 116: Must be in 8087 mode to compile this` will be generated. To make sure that your program compiles exactly like on your computer, in *Borland Pascal 7.0* press `Ctrl+O,O`. This will include in your source files all the compiler directives with the options from your environment and you may change them as you wish. Do not worry if you do not know what some of them mean, they will cause no harm and if the program runs properly on your computer then it will do so on the computer used for evaluation.

All the programs should read the input data from the specified input file. No data should be expected from the standard input (keyboard). If your program waits for a reaction from the keyboard then it will be halted after the time expires and you will obtain no points for your solution, even if it is correct. So, please do not include in your code sequences like `Readln`, `getch()`, `getche()` etc.

All the programs should write the output data to the specified output file. You may write data to the standard output but this will only slow down your program, so you are not advised to do so. All data written to the standard output is ignored during the evaluation process. The evaluation program will check only the solution generated to the specified output file.

The input file is always in the current directory, so do not try to read input data from other locations. The output file should also be generated in the current directory, so do not write output data in any other location.

You must always use the specified names for your source, input and output file. Using other names will cause your program to run improperly.

Do not submit solutions after the deadline (the last day of every month)!

Example

At each round there will be three tasks to solve. Suppose one of the task descriptions is the following:

Source file name: **`SUM.PAS`**
Input file name: **`SUM.IN`**
Output file name: **`SUM.OUT`**
Description: Given two numbers find their sum.
Input data: The input file contains, on a single row, the two numbers separated by a single blank character.
Output data: The output file should contain the sum of the two numbers.

Example:

`SUM.IN`	**`SUM.OUT`**
2 4	6

Maximum execution time: 1 second/test

After reading the task description you start solving the task. You notice that it is not clear whether the numbers are integer or real numbers. To clarify this you must submit a question. You may not ask the question: *Are the numbers integer or real?* because this is not a YES/NO type of question. A solution is to ask: *Are the number integer?*. You will get the answer NO and you will know that the numbers are real. Now a big question arises: *What precision should I use for writing the result?* It looks like the *GInfo Editorial Staff* forgot to tell you this. Of course you will have to ask a YES/NO question, so you may try *Is a three decimal precision sufficient?* The answer to your question will be GENERAL ANNOUNCEMENT and will be followed by another e-mail containing something like:

The result must be written with a precision of two decimals. We apologize for the incomplete task description!

Hopefully we will never have to send such e-mails, but we are only humans.

After getting the general announcement you may start writing your code. A correct solution might be:

```
 1: Program SUM;
 2: var a,b:Real;
 3:     f:Text;
 4: Begin
 5:   Assign(f,'SUM.IN');
 6:   Reset(f);
 7:   Readln(f,a,b);
 8:   Close(f);
 9:   Assign(f,'SUM.OUT');
10:   Rewrite(f);
11:   Writeln(f,a+b:0:2);
12:   Close(f)
13: End.
```

Please note that the line numbers are not part of the program and are provided for reference purposes only.

If you add a line `1': uses Crt;` and a line `4': Clrscr;` then you're going to get a nice `Error 200: Division by zero`, as explained in the advice section.

If you add a line `12': Readln;` the evaluation result will be `Time out!` because the program will not terminate, waiting instead for the <**ENTER**> key to be pressed and it will be halted after one second.

If you use the line `5: Assign(f,'C:\SUM.IN');` then your program does not work properly unless the current directory is the root of the drive `C:` (which most likely won't be the case). Otherwise you will only get a `File not found` error.

If you use the line `5: Assign(f,'MYFILE');` then your program will not read input data from the specified input file and will run improperly.

If you use the line `9: Assign(f,'C:\SUM.OUT');` then your program will seem to work properly, but the evaluation program will find your output file only if the current directory is the root of drive `C:` (again, chances for this to happen are very low) Otherwise the evaluation program will return `No output file`.

If you use the line `9: Assign(f,'MYFILE');` then your program will not write to the specified output file and the evaluation program will return `No output file`.

If you forget about line `12`, then the output file will not be closed and the evaluation program will return `No data written to the output file`.

If you use the line `7: Readln(a,b)` then your program expects input data from the standard input and will be halted after one second. You may use this line only if you have a sequence like

```
5: Assign(Input,'SUM.IN');
6: Reset(Input);
7: Readln(a,b);
8: Close(Input);
```

If you use the line `11: Writeln(a+b:0:2);` the output data will be written to the screen and not to the specified output file. You may use this line only if you have a sequence like

```
9:  Assign(Output,'SUM.OUT');
10: Rewrite(Output);
11: Writeln(Output,a+b:0:2);
12: Close(Output)
```

If you use the line `7: Readln(a); Readln(b);` then your program will read only the first number from the input file and will not read correctly the second number. Consequently, it will not generate the correct result.

These are the most common errors that appeared at the first edition of our contest. Hopefully, you will not make the same mistakes this year.

After writing the source code you must submit the solution, along with the solution for the other two tasks. Suppose that the source files for the other two tasks are **DIF.PAS** and **MUL.PAS**. You must now create the archive that has to be submitted. This archive should contain a directory named `yyyy` where `yyyy` is your registration code (for example `2457`). You must first create a directory called `2457` then copy the files **SUM.PAS**, **DIF.PAS** and **MUL.PAS** inside this directory. You must now create the **.bat** file we shall use for compiling your programs; it will be called **ROUND1.BAT** and it will look like this:

```
bpc sum.pas
bpc dif.pas
bpc mul.pas
```

Use `bpc` for compiling *Pascal* sources and `bcc` for compiling *C/C++* sources.

Having created this file you are ready to build the archive and submit it. Use a packer that creates .zip or .rar archives; do not build a self-extracting archive. The name of your archive file should be **2457R1.RAR** or **2457R1.ZIP**.

Now you may send us the e-mail containing your solutions. Do not forget to attach the archive and to write in the *Subject* line: `2457 Round 1`.

Attention! Do not use the `2457` registration code unless it is really your *APC* ID. Use your own ID, here `2457` is used only as an example!

SETI@home

Many people believe that there is life in the outer space, but nobody could ever prove that we are not alone in the Universe. There are several scientific projects that try to determine if there really is intelligent life in the extraterrestrial space. One of this scientific efforts is SETI (Search for ExtraTerrestrial Intelligence); this article describes the way you can help this project.

There are many methods used for searching extraterrestrial intelligent life; many of them search billions of radio frequencies that flood the Universe looking for signals that may have been transmitted by intelligent beings. Another method is to look for signals in pulses of light emanating from the stars.

One of the SETI projects is SETI@home; this project allows anyone possessing a computer and an Internet connection to take part in the search. The computer is used only when the user is not present, so his/her activity is not disturbed. A screen-saver is installed and, while this screen-saver is running, the computer searches for extraterrestrial intelligence.

About SETI

Most of you have surely seen movies like *Star Trek*, *Star Wars*, *Alien*, *Independence Day* etc. All of them deal with alien civilizations and their relationships with human beings. Sometimes the members of alien societies are friendly, other times they are hostile. A predominant characteristic is that the aliens are humanoids. Many people hope that, in the near future, a contact between humanity and an alien society will be made. There will be many things about each other to be learned and discovered. SETI is one of the projects that try to make this moment possible and your help would be appreciated.

If we assume that aliens are trying to contact our race then the least we can do is looking for their signals. There are several programs that try to find evidence of life elsewhere in the cosmos. Collectively these programs are called SETI (*Search for ExtraTerrestrial Intelligence*).

Alien civilizations

Our galaxy has more than 400.000.000.000 stars, our *Sun* being only one of them. The Universe has billions of galaxies and our *Milky Way* is only one of them. It looks like there could be many intelligent creatures out there! We shall try to make an initial estimate for the number of alien civilizations. The first who tried such an approximation was the astronomer *Frank Drake*. A simple equation, now called *Drake Equation*, maps out the possibilities. This equation is quite easy to understand, so you don't need to be a genius in order to read the following paragraphs. The formula is:

```
N=R*f(p)*n(e)*f(l)*f(i)*f(c)*L
```

We shall now explain the variables that appear in the previous equation.

- `N` represents the number of communicating civilizations in the *Milky Way* galaxy
- `R` represents the rate of "suitable" star formation in the galaxy
- `f(p)` is the fraction of stars that have planets
- `n(e)` is the number of these planets around any star within the suitable ecosphere of the star; an ecosphere is a shell that surrounds a star within which the conditions are suitable for life to form
- `f(l)` is the fraction of those planets within the ecosphere on which life actually evolves
- `f(i)` is the fraction of those planets on which intelligent life evolves
- `f(c)` is the fraction of those planets where intelligent life develops a technology and attempts communication
- `L` is the length of time that an intelligent, communicating civilization lasts

In what follows we shall look at each of these factors and try to assign them some reasonable values.

Star formation

Although the rate of suitable star formation was much higher when our galaxy was formed, one can see that stars are being born today, too. A good guess, at the rate of this star formation, is 20 stars/year, so we take `R=20`.

Stars with planets

Stars are formed by huge collapsing gas clouds. Many of these clouds have a bit of rotation; as they collapse they spin faster and faster and a flattened disk of gas is formed. In the center the main star will appear, and out further smaller eddies can form planets. Until recently there was no evidence that there are planetary systems around

stars other than the *Sun*. During the last years several teams of astronomers have announced the discovery of planets surrounding nearby stars. This extraordinary discovery increased the chances for planets to exist around many other stars. We may estimate that one half of the stars form planetary systems and the other half form binary star systems. Hence we may consider the value `f(p)=0.5`

Planets in the ecosphere

The `n(e)` factor is a little tricky, so finding an appropriate value for it might be difficult.

Small stars are cool and red; their planets should orbit very close to be in the ecosphere. Their ecospheres are very narrow, so there is not much room for planets inside them. Moreover, planets that orbit very close to their star are often tidally locked and present only one face to the star at all times (the situation is similar with the *Moon* and the *Earth*). The atmosphere of such a planet would freeze on the cool side that faces away from the star; this conditions do not promote life.

On the other hand, huge hot blue stars have a farther and wider ecosphere, but judging from our solar system, planets are spaced further apart the farther they are from the star, so the wider ecosphere is cancelled by this effect. These large stars also burn their fuel faster, so they don't last very long. Usually these stars become novas or supernovas before life gets a chance to appear.

Our *Sun* is an average-sized star and there are two (*Earth* and *Mars*) or maybe three (+*Venus*) planets within the ecosphere.

A conservative guess for the number of planets within the 'life zone" is one, so `n(e)=1`.

Life on planets

The main problem with the `f(l)` factor is that we have only a few examples of planets where conditions are proper for life to evolve. As stated before, it is possible that *Venus*, *Earth* and *Mars* could all have had proper conditions. We know that life evolved on *Earth* and there is now tantalizing evidence for primitive life existing on *Mars* billions of years ago. A conservative guess for this number is `0.2`, or, in other words, on one in five planets with proper conditions life will evolve. `f(l)=0.2`.

Intelligent life

It is very difficult to say how many of these planets will evolve intelligent life. If we really believe in the evidence of natural selection and survival of the fittest, then it looks like there are `100%` chances for intelligent life to appear - intelligent life is a natural outcome of evolution. The only example we have is the *Earth*, so we may assume `f(i)=1`.

Technologic societies

We have to see how many of the intelligent species will develop technology and use it to communicate. On *Earth*, we see that humans do this, but we also see that whales and dolphins did not develop technology even if they have a moderate level of intelligence. We shall set the factor `f(c)` to `0.5` as a first guess.

Self destruction

The hardest factor to be determined is `L`; it indicates the number of years that a technologically adept and communicative civilization lasts. We've only been in this phase of our evolution for about `50` years. Do advanced civilizations blow themselves up after discovering the technology to do so? Or do they get together and solve their problems before this happens? For now, let's not assign a number to `L`.

Final Result

Let's plug in the numbers and see what we get.

```
N=R*f(p)*n(e)*f(l)*f(i)* f(c)*L
N=20*0.5*1*0.2*1*0.5*L
```

We obtain `N=L`, so the number of intelligent communicating civilizations in the galaxy equals the number of years such a civilization would last!

Conclusions

Most scientists hope that if a civilization can overcome its initial tendency to destroy itself with its own technology, then that civilization is likely to last for a very long time. Let's hope those scientists are right. In any case, there should be at least `50` (the number of years we've been around communicating) and if a communicative civilization lasts for millions of years, there may be possibly millions of civilizations we can look for.

The values used for the calculations are provided by the SETI Staff. We believe that several values are artificially increased. For example, we think that SETI exagerates when claiming that `f(l)=0.2`. Probably, the real value is very close to zero. Anyway, nobody can say there is no possibility for extraterrestrial communicating intelligent life to exist. We hope that we convinced you that the SETI programs are not a waste of time.

About SETI@home

The data provided by a telescope must be processed in order to check for signs of extraterrestrial intelligence. Large computers have been built for this process, but because of the large amount of data that has to be processed, these computers do not look very deeply at the data; they do not look for weak signals or certain signal types. The reason for this is the limited amount of computer power available for data analysis. The SETI team discovered that there are thousands of computers that are available for use, but nobody has realized this fact yet. Many computer screens show most of the time only a screen saver, accomplishing nothing and wasting electricity. This is where *SETI@home* comes into the picture. The *SETI@home* project hopes to convince you to allow SETI to borrow your computer when you aren't using it and to help them "*...search out new life and new civilizations.*" They will do this with a screen saver that can go get a chunk of data over the Internet, analyze that data, and then report the results back to SETI. When you need your computer back, the screen saver instantly gets out of the way and continues its analysis only when you are finished with your work.

It's an interesting and difficult task. There's so much data to analyze that it seems impossible! Fortunately, the data analysis task can be easily broken up into little pieces that can all be worked on separately and in parallel. None of the pieces depends on the other ones. Also, there is only a finite amount of the sky that can be seen from *Arecibo* (the telescope used for receiving extraterrestrial signals). In the next two years the entire sky as seen from the telescope, will be scanned three times. SETI believes that it will be enough for this project. By the time they have looked at the sky three times, there will be new telescopes, new experiments, and new approaches to SETI. We hope that you will be able to participate at them too!

Work units

Data is recorded on high-density tapes at the *Arecibo* telescope in *Puerto Rico*, filling about one 35 GB DLT tape per day. Because *Arecibo* does not have a high bandwidth Internet connection, the data tape must go by snail-mail to *Berkeley*. The data is then divided into 0.25 MB chunks (called "work-units"). These are sent from the *Seti@home* server over the Internet to people around the world to analyze.

SETI@home looks at 2.5 MHz of data, centered at 1420 MHz. This still is a too broad spectrum to send it to users for analysis, so they break up this spectrum space into 256 pieces, each 10 KHz wide (actually 9766 Hz). This is done with a software program called the "*splitter*". These 10 KHz pieces are now more manageable in size. To record signals up to 10 KHz one has to record the bits at 20000 bits per second (bps). (This is called the *Nyquist frequency*.) SETI sends you about 107 seconds of this 10 KHz (20kbps) data. 100 seconds times 20000 bits equals 2000000 bits, or about 0.25 MB. Additional information about the work-unit is sent, so the total comes out to about 340 KB of data.

Transferring data

SETI@home connects only when transferring data. This occurs only when the screen saver has finished analyzing the work-unit and wants to send back the results (and get another work-unit.) The connection is established only with the user's permission and the user is allowed to control when his/her computer connects to SETI. There also is the option to set the screen saver to transfer data automatically as soon as it's done with the current work-unit. The data transmission lasts for less than 5 minutes with most common modems.

SETI keeps track of the work-units using a large database. When the work-units are returned to SETI, they are merged back into the database and marked "*done*". The computers look for a new work-unit for the user to process and send it out, marking it "*in progress*" in the database.

Science

Now let's see what the *SETI@home* application actually does. First let us see what we expect extraterrestrials to send. We expect that they would want to send us a signal in the most efficient manner for them, that would allow us to easily detect the message. Now, it turns out that sending a message on many frequencies is not efficient as it takes a lot of power. If one concentrates the power of the message into a very narrow frequency range (narrow bandwidth) the signal is easier to weed out from the background noise. This is especially important since we assume that they are far enough away that their signal will be very weak by the time it gets to us. So, we're not looking for a broadband signal (spread over many frequencies), we're looking for a very specific frequency message. The *SETI@home* screen saver acts like tuning your radio set to various channels, and looking at the signal strength meter. If the strength meter goes up, that gets SETI's attention.

Another factor that helps reject local (earth-based and satellite-based) signals is that local sources are more or less constant. They maintain their intensity over time. On the other hand, the *Arecibo* telescope is fixed in position. When *SETI@home* is in operation, the telescope does not track the stars. Because of this, the sky "*drifts*" past the focus of the telescope. It takes about 12 seconds for a target to cross the focus (or "*target beam*") of the dish. We therefore expect an extraterrestrial signal to get louder and then softer over a 12 second period. Since we are looking for this 12 second "*gaussian*" signal, SETI sends you about 100 seconds of data. Also, SETI allows the data in the work-units to overlap a little so we won't miss an important signal by cutting it off early in the analysis.

Computations

The SETI@home software searches for signals about 10 times weaker than the *SERENDIP IV* search at *Arecibo*, because it makes use of a computationally intensive algorithm called "*coherent integration*". No one else (including the *SERENDIP* program) has had the computing power to implement this method. Your computer performs *Fast Fourier Transforms* on the data, looking for strong signals at various combinations of frequency, bandwidth, and chirp rates.

We shall now present you the steps which are taken on each of the work-units you get from SETI. The most computationally intensive portion of the calculation is to "*de-chirp*" the data - that is, to remove all the effects of the *Doppler* acceleration. At the finest resolution, this has to be done for 5000 times, from -5 Hz/sec to +5 Hz/sec in steps of 0.002 Hz/sec. At each chirp-rate, the 107 seconds of data is de-chirped and then divided into 8 blocks of 13.375 seconds each. Each 13.375 second block is then examined with a bandwidth of 0.07 Hz for peaks (that is 131072 tests (frequencies) per block per chirp rate!) In this first step, your computer does about 100,000,000,000 calculations!

The other bandwidths have to be tested too; the next step doubles the bandwidth to 0.15 Hz. Starting at this bandwidth, the chirp range is doubled (the chirp rates are from -10 Hz/sec to +10 Hz/sec). Even

though this doubles the range, the computer only have to examine 1/4 the number of rates due to the increase in bandwidth. So we have twice the chirp range times 1/4 the number of chirps. We end up doing about 1/2 the amount of work we did above at the highest resolution narrow bandwidth, or about 50,000,000,000 calculations.

The next step doubles the bandwidth again (from 0.15 to 0.3 Hz) and reduces again the chirps by 1/4. (The -10 Hz/sec to +10 Hz/sec chirp range is maintained for the rest of the calculations.) This step (and all successive steps) take 1/4 the calculation of the previous step. In this case only 12,500,000,000 calculations. This continues for a total of 14 doublings of bandwidth (0.07, 0.15, 0.3, 0.6, 1.2, 2.5, 5, 10, 20, 40, 75, 150, 300, 600, and 1200 Hz) to bring you to a grand total of slightly more than 175,000,000,000 operations on the 107 seconds of data. As you can see most of the work is done at the narrowest bandwidth (about 70% of the work).

Finally, signals that show a strong power at some particular combination of frequency, bandwidth and chirp are subject to a test for terrestrial interference. Only if the power rises and then falls over a 12 second period (the time it takes the telescope to pass a spot in the sky) can the signal be tentatively considered extra-terrestrial in nature.

An average, current model home computer should take between 10 and 50 hours to complete one work-unit. This assumes that the computer ONLY works on *SETI@home*.

The SETI@home app.

In the previous section we said many things about the *SETI@home* application, but there are several other things we must mention.

First of all, *SETI@home* may run as a normal application, not only as a screen saver. Moreover, you may run in background *SETI@home* all the time, but you must be aware of the fact that this application consumes 15 MB of your memory. Users possessing computers having less than 64 MB of memory should not use this facility.

For becoming a *SETI@home* user you must install the application and then register yourself as a user. The application may be downloaded from the *SETI@home* site (http://setiathome.berkeley.edu); the file size is 771.5 KB.

After installing you may register yourself; start the application and from the *Settings* menu choose *Create/Change Login...*, then press the *Create New Account* button; fill in the form and then press the *Submit* button; you will be registered as a new *SETI@home* user and you will receive your first work unit for processing.

Statistics

Maybe the most interesting part of the whole *SETI@home* project consists of the various statistics available. There are statistics for each country, for groups or individuals; all are available on the *SETI@home* site.

The countries are ranked according to various criteria; we shall now see what positions are occupied by the CEOI participants (at *August 12th, 2000*). The criteria we have chosen are the number of data units (results) processed by users from that country, the number of users from that country, the amount of CPU time used by computers in that country and the ratio between the CPU time and the numbers of results submitted to SETI.

As of *August 12th, 2000* there were more than 2,200,000 SETI users and more than 168,000,000 data units were processed; the average number of data units processed by one user was about 75.33.

The GInfo group

As we mentioned before, users may form groups to make more interesting statistics. We hope that you are interested in this *SETI@home* project and you will soon become a *SETI@home* user (if you are not already). Even if you are a new or old user, we hope that you will join the *GInfo* group. For joining our group you must enter the *Groups* page on the *SETI@home* site, search for the *GInfo* group and then click *Join*.

Conclusions

We do not try to convince you that extraterrestrial intelligence really exists, we just want to say that sometimes your computer does nothing even if it is turned on and its processing power might be used for more important activities. *SETI@home* is only one (the most well-known) of the projects that borrow people's computers for scientific activities. So, if you do not mind wasting a few minutes for receiving data units, your computer will become more useful to the humanity.

Country	Results	Users	CPU Time	CPU Time/Results
CRO	39th	39th	36th	154th
CZE	15th	17th	13th	110th
GER	2nd	3rd	4th	57th
HUN	25th	29th	24th	82nd
POL	14th	14th	15th	46th
ROM	45th	48th	45th	58th
SVK	36th	44th	35th	99th
SLO	31st	43rd	32nd	90th
MOL	117th	132nd	127th	24th
HOL	6th	5th	6th	91st
USA	1st	1st	1st	150th

Tabel 1

The processors WAR (INTEL vs. AMD)

The day of March 6th 2000 will remain in the history of the IT&C industry as the day when the first normally cooled 1 GHz PC microprocesser was launched. The company that first crossed the 1 GHz line was Advanced Micro Devices, better known as AMD. Archrivals Intel Corp. launched their own 1 GHz Pentium III processer two days later. This is the peak of a processer war started many years before...

Just one year ago, the processor industry was dominated categorically by one company: *Intel Corporation*. They offered the fastest processors, the most innovative solutions and were widely regarded as the company to go for if you wanted a professional system. The only other two companies were seen as some kind of pale imitators, unable to come up with something original, and only good for being used in a cheap system where cost was a primary concern. Many people considered the two small companies had no chance of surviving *Intel*'s ruthless domination. With *Cyrix* they are partially right, in spite of being acquired by *Via* last year they are still unable to return to the market in full force. But those who predicted *AMD*'s dissapearance couldn't have been more wrong.

True, history offers many examples of (often) very powerful companies trying to make it in the processors' world. In the days when a 386 was state of the art, there were several companies producing 386 clones. Besides Intel producing the "original" 386, there was *AMD*, *Cyrix*, *IBM* (with a 386 named *Blue Lighting*), *Texas Instruments* (with *ti386*) and many others. But, although resembling very much one another, the *Intel* processors were the fastest (and most expensive, at the same time). It was around those times that *Intel* introduced their famous and very successful "*Intel inside*" promotional campaign. And they slowly began to dominate the market, and in 1993 they introduced the *Pentium* processor, named this way (instead of 586 - the logical name that would have followed after 486) because one can not protect by trademark a name formed by numbers. By then, *IBM* and *Texas Instruments* had long exited the microprocessors market, and *Cyrix* and *AMD* looked set to follow their example.

A very important part in *AMD*'s future turnaround was played by a small company named *Nexgen*. It was one of the many companies trying to make a succesful Intel clone, but they truly differentiated from the others. They had a small but very talented team of engineers and their processors incorporated many design features found only in much more expensive types of processors belonging to a totally different class of computers, the very expensive professional workstation computers. More precisely, they were the first to bring RISC technology to the x86 family.

With the *K5* processor, *AMD* gave a pale response to *Intel*'s very successful and very well received *Pentium*, but with the *K6* things changed considerably. The acquisition of *NexGen* began to show its fruits, the *K6* incorporating many features from the *NexGen* products, and the *NexGen* team integrating very well into *AMD* structures and playing an active role in the development of new processors. *K6* wasn't able to surpass the performance of *Intel*'s processors (suffering a performance hit mainly in the floating-point intensive applications like CAD, 3D modelling and very important, games). Still it offered good performance and excellent value for money. It did well in sales and was well received by the press.

The turning point: Athlon

In the mid-90's, when *Intel* seemed to have no rival in the market, ***Andy Grove***, then *Intel*'s chairman, uttered some words which would become famous: "Only the paranoid survive". And he was right indeed, because 1999 was to be a turning point in *AMD*'s history: they launched their *Athlon* processor.

The *Athlon* processor is important because it is the first *AMD* processor designed from the ground up as a fast-performance processor, instead of being a mere *Intel* clone. Incidentally, it used many features from the former *NexGen* designs. It was also the first

processor to outperform Intel's fastest processors on a clock by clock basis and also the first *AMD* processor to run at clock frequencies higher than *Intel*'s top frequencies: at its launch, it was available at 500, 550, 600 and 650 MHz speeds, while the fastest *Pentium* ran at 600 MHz.

The *Athlon* sold remarkably well, despite several problems it had at launch. First, the lack of motherboard support, due to a number of factors. When it designed the *Athlon*, *AMD* never intended to provide the chipsets for it. It was expected that *VIA* and the other chipset manufacturers would be able to present final products before the official launch, but at launch the *VIA* chipset was not ready, and *AMD* were forced to supply their 750 chipset as a solution, a fact they weren't prepared for. The 750 chipset was meant as a test chipset and it lacked many features found in other chipsets like AGP 4x. Then *Intel* pressures made many manufacturers cancel *Athlon* motherboards plans, when faced with threats of not getting any more chipset supplies from *Intel*. A good example was the well-known motherboard manufacturer *ASUS* who although designed an excellent Athlon motherboard sold it only to OEMs and did not even have information about it for months. The boards came packaged in white boxes instead of the colorful *ASUS* ones and did not have the *ASUS* name anywhere on them. Even the BIOS updates were only available from a ftp site in *Germany* and not from *ASUS* official site.

Another factor against *Athlon* was that the big manufacturers didn't offer *Athlon* machines from the beginning, leaving users with two choices: to buy from smaller manufacturers or build their own. Still the *Athlon* sold well, with demand coming mainly from power-hungry gaming enthusiasts and power users who needed a lot of computing power.

Chipsets from 3rd party manufacturers began to appear under the form of *VIA KX 133* (and later *VIA KT 133*) and this, together with *Intel* supply problems slowly made Athlon a popular solution.

Companies soon began to offer faster processors at lower prices and this quickly led to...

The Race to 1 GHz

The 1GHz value has no special technical significance, it is more a psychological number, and therefore being the first to release a 1 GHz processor presented significant marketing attraction. The spectacular race towards the psychological 1 GHz barrier ended in March, its two contenders crossing the finish line almost simultaneously. The race ended sooner than expected; initially *Intel* and *AMD* announced that their Giga-processors will be available only in the late 2000 or early 2001. Competition made the two companies launch these processors earlier, much to the user's delight, as this meant lower prices for the other processors.

AMD Athlon 1000

On *March 6th, 2000* AMD launched the first ever PC processor that works at a 1000 MHz frequency. On this occasion ***Mr. W.J. Sanders III***, chairman and chief executive officer at *AMD* declared that reaching the 1 GHz barrier was, for a long time, one of the main purposes of the PC processors producers. In his opiniom, the launch of a processor capable of executing one billion clock cycles per second is, in our industry, an event equivalent to breaking the sound barrier. Similar to the way in which ***Chuck Yeager***'s performance marked the beginning of a new era in aviation, the 1 GHz processor anticipates a new era in the information technology.

Because it wasn't a secret that *Intel* would launch, a couple of days later, their *Giga-Pentium III*, Sanders added that everybody knows that ***Neil Armstrong*** was the first man walking on the *Moon*, ***Roger Bannister*** was the first man running a mile in less than four minutes or *Edmund Hillary* was the first to conquer the *Everest* and nobody remembers who were the ones who did these things for the second time.

The *AMD Athlon* processor was launched only nine months after the release of the first Athlon processor working at 500 MHz. This fact contradicts the famous Moore law which claims that processor performances will be doubled every 18 months.

The first companies that released systems using *Giga-Athlons* were *Compaq* and *Gateway*.

Together with the *AMD Athlon 1000*, the *AMD Athlon 900* and *AMD Athlon 950* processors were launched. *AMD* wanted to fill the gap between the A*MD Athlon 850* (their best processor before this launch) and the new *Giga-Athlon*.

At launch time a *Giga-Athlon* could have been bought for 1299$. Now it is available for less than 500$.

Intel Pentium III 1000

Intel didn't take long to reply; the Intel *Pentium III* processor working at 1 GHz was launched on *March 8th, 2000*. On this occasion ***George Alfs***, spokesman for *Intel*, said that this processor is aimed at those who could really appreciate its qualities, like enthusiast users or gamers. During the second quarter of this year the number of available *Giga-Pentium III*'s was limited. It looks like starting with *August 14th*, the *1 GHz Pentium III* may be bought by anyone who can afford it.

It is obvious that *Intel* was not ready for this launch; the company had to jump directly from 800 to 1000 MHz and was not able to make intermediate versions available on the market. After two weeks *Intel* began to fill in the gap by launching *Pentium III* processors working at 850 and 866 MHz. The *Intel Pentium III 933* became available in *June*.

Athlon vs. Pentium III

At launch time, the *Giga-Pentium III* was cheaper than the *Giga-Athlon*. Anyway, because of the limited quantities, it was very difficult to get a *Pentium III 1 GHz* processor. In fact, you could only buy entire systems from *"tier 1"* manufacturers like *Dell* or *IBM*. It was impossible to buy a *Giga-Pentium III* from retail stores. An *AMD* fan said: "You could buy 1 GHz processors from Intel, too, if there were 1000 such processors in the entire world!". There were also funnier comments; many remembered the following joke:

A lady enters a store and wants to buy a pound of apples. The man in the store brings the apples and says: "99¢, please". Very surprised, the lady replies: "I can buy apples from the super-market for 79¢". The man in the store asks: "Then why don't you go and buy apples from the supermarket?". Very seriously, the lady answers: "I was at the super-market, but they are out of stock". Smiling, the vendor says: "Lady, when I'm out of stock, I sell apples for 29¢".

The fact is that the number of available *Giga-Pentium III*'s is indeed very low. Availability has improved slightly only recently, more than 4 months after its paper launch. Meanwhile the *Giga-Athlons* are and have been widely available, not only in systems from big (*"tier 1"*) manufacturers but also in systems from smaller manufacturers as well as in the form of separate components that do not force you to buy an entire system.

Performance-wise, it seems that *Pentium III 1000* is slightly superior to an *Athlon 1000* in office applications. This is due to the fact that the new *Pentium III Coppermine* has 256K of integrated cache that runs at the processor frequency, while Athlons have 512K that only runs at 1/2 (or even 1/3 - for the 1 GHz part) of the processor speed. (This situation has changed recently with the introduction of the new *Athlon* with the *Thunderbird* core which is very similar to the *Coppermine* core from the cache perspective - more on this later). Still the *Athlons* are and have been clearly superior in scientific applications which require lots of floating point operations. (Probably the best known such scientific application is *Quake III Arena*.)

Very recently *AMD* has introduced a new version of its popular Athlon processor known as *Athlon Thunderbird*. The *Thunderbird* eliminates the cache handicap and is once again more than able to compete head to head with the fastest *Intel* processors. Besides, *Athlon*'s architecture is much more advanced, leaving a lot of room for improvement, while the *Pentium III*, according to *Intel* itself is very near to its architectural capabilities, using essentially the same architecture (vastly improved, of course) as the 1994 *Pentium Pro*. (But let's not forget that the next generation of the *Pentium* series known as *Wilamette* will be launched in the late 2000 under the name of *Pentium 4*.) *Intel* still doesn't seem to have learned its lesson and has recently (*July 31st*) released the *1.13 GHz Pentium III*. Again it looks like another paper launch to us, and moreover one plagued by technical and reliability problems (see www.tomshardware.com), something previously unheard of from the maker of the "world's most reliable processors". *AMD* is expected to

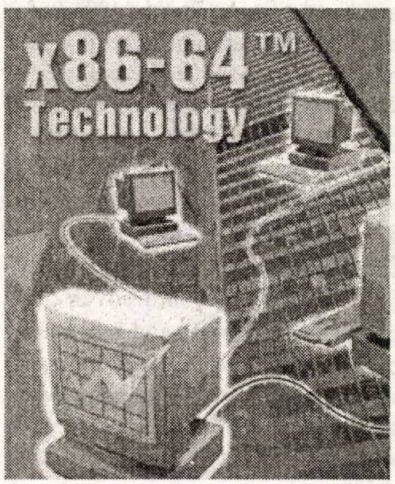
launch its *1.1 GHz Athlon* very soon, so the battle is open again. Anyway, it seems that at least in the short term *AMD* will be the leader in the processor field.

Duron vs. Celeron

Together with the *Thunderbird* AMD also launched the *Duron* processor, aimed at their traditional, low-cost market. The *Duron* will compete directly with *Intel*'s *Celeron* processor, being very similar to it. The *Duron* is based on the *Thunderbird* core, just like the *Celeron* is based on the *Pentium III* core. The Duron also has a reduced level 2 cache size. But the similarities stop here. The *Duron* runs at a FSB of 200 MHz, just like its older brother, unlike the *Celeron* which is artificially limited to a low 66 MHz bus. The *Duron* memory runs at 133 MHz, while the *Celeron* runs its memory at 66 MHz. Given these facts, it is no surprise that the *Duron* clearly outperforms the *Celeron* at the same clock frequency. Add to this the fact that the *Duron* is also cheaper than the *Celeron* and it looks like *AMD* will continue to dominate the low-cost market.

The mobile market

Another confrontation will take place on the mobile market. *AMD* already dominates this market (when taken into account the number of processors installed in notebooks), their mobile *K6-2+* being very well received. But *Intel* always was the performance king, their *mobile Pentium III* being faster than the *mobile K6-2+* and *mobile K6-III+*.



But *AMD*'s new mobile processor based on the *Athlon* core looks set to change this. But an even more important issue will be the new power-saving features which will be introduced by the two companies in their processors. Remember that performance is not the most important consideration in mobile market, but rather battery life. *Intel* announced their *SpeedStep* technology that will allow you to use your processor at a higher frequency when operated connected to a wall plug and to switch automatically to a lower frequency when operated on batteries. But *AMD* looks superior here too, because their PowerNow! technology which will be introduced shortly will also offer an additional feature over *Intel*'s technology: it will allow you to automatically increase/decrease the frequency in several steps based on the current load of your notebook. In other words, it will run at a lower frequency when you're doing simple tasks like reading your e-mail but will automatically increse the frequency when you're doing something more computationally-intensive like querying a customer database, for example.

64-bit confrontation

But another battle is appearing on the horizon, which could prove decisive in the battle for tomorrow's microprocessors: the fight to impose the 64 bit architecture of the next generation's processors.

It is interesting to note that this will be the first time the two architectures from the two companies will separate, resulting in systems incompatible with each other.

Intel will use a radical approach for its new 64 bit architecture. The next *Intel* processor will be named *Itanium* and will use an architecture known as EPIC (*Explicitly Parallel Instruction word Computer*) developed in collaboration with *Hewlett-Packard*. It will be totally incompatible with the existing 32 bit applications, so to take real advantage of the new processor you will need new operating systems and applications. Actually, the *Itanium* will be able to run existing 32 bit applications but in a special emulating mode which will be much slower than today's *Pentium III* processors. This is reminiscent of the 1994 *Pentium Pro* processor which had excellent 32 bit performance but ran 16-bit applications slower than the much cheaper *Pentiums*.

AMD has taken a different approach. Its *Sledgehammer* processor will run 32 bit applications natively, and its x86-XA (*eXtended Architecture*) will be an evolution of the present 32 bit one. This is just like the 386 processors that although had native 32-bit capability were also able to run 16-bit applications very fast, in fact faster than the previous ones like 286 and 8086.

Opinions among analysts are split as to what approach will prevail in the end. A very important issue here is software support, an area in which Intel seems to have an advantage. Major players in the industry have announced support for the *Itanium* platform with firm commitments for 64-bit operating systems: *Sun* (*Solaris*), *Microsoft* (*Windows*), *IBM/SCO* (*Project Monterey* - a 64-bit UNIX operating system). And of course the open-source *Linux* community, which announced support for both platforms. On the other side, *AMD* has no major player by its side apart from the *Linux* community. *Microsoft* remained quiet over plans for a 64-bit version of *Windows* for *Sledgehammer*, although one senior executive at *Microsoft* said that "*AMD* has offered us numerous details about their 64 bit plans" and that they (*Microsoft*) "provided some feedback". Of course, *AMD* is able to run all of the current operating systems but to be a credible alternative it must have native 64 bit operating system support. On the other hand, *AMD* has a huge advantage in that it is able to run current 32-bit applications natively, because the quick transition to 64 bits that *Intel* would prefer is *very* unlikely. This is because it all depends on the existence of good, mature compilers for its 64 bit architectures. But all insiders from the compiler developers agree that good developing good com-

pilers for a completely new architecture is not a science but a form of art, and that such compilers take many years to mature.

The *Sledghammer* is very important for *AMD* because it will allow them to enter the high-end end mutiprocessor server market. This will allow them to enter the huge corporate market, currently strongly dominated by *Intel*. *AMD* hasn't had much success on this market because very often, large companies want to have the same solution for their computing needs, starting from notebooks and all the way to large departamental servers. *AMD* has unfortunately been absent from the server market and thus was not able to compete in this area. With the *Sledgehammer* this may change soon.

To address the same problem mentioned above *AMD* will introduce ahead of *Sledgehammer* another version of its *Athlon* core known as *Mustang*, incorporating larger amounts of level 2 *cache* (up to 2 MB). *Mustang* will compete head to head with *Intel*'s *Xeon* in the server market.

Memory

We can not go over this heated dispute without looking at a very important issue: memory. Yes, memory, because as the new processors get faster and faster memory becomes a major bottleneck. Analysts agree that one of *Intel*'s huge mistakes was stubbornly supporting a new memory standard called *Rambus Direct RAM* (RDRAM). An interesting technology in the beginning, which promised a great improvement in performance at an affordable price, *Rambus* quickly began to have adversaries. First of all it was very expensive, at one moment *Rambus* memory being 8 times more expensive than equivalent SDRAM. Second, although *Intel* and *Rambus* benchmarks proved increases in performance, independent reviewers proved that not only did *Rambus* offer no noticeably better performance, it also made computers run slower in many applications. And, very importantly, *Rambus*-based systems were faced with a series of technical difficulties to start with. A few days before the official launch of the *Intel* 820 chipset *Intel* engineers discovered a serious bug in the chipset in the RDRAM controller.

More specifically they discovered that serious problems arose when the maximum of three memory modules were used. Not having time to redesign the chipset, they opted for an artificial solution limiting the maximum number of memory modules to two. Later more problems were found culminating in a recall of all *Intel 820 Capecod* motherboards.

Rambus used all kinds of dirty tricks to put Rambus at an advantage in benchmarks. They used only the most expensive PC800 RDRAM when the majority of systems were equipped with slower PC600 and PC700 RDRAM, they used the slower PC100 SDRAM and not the PC133 SDRAM widely available and they used a specially modified BIOS with an extra latency cycle added artificially.

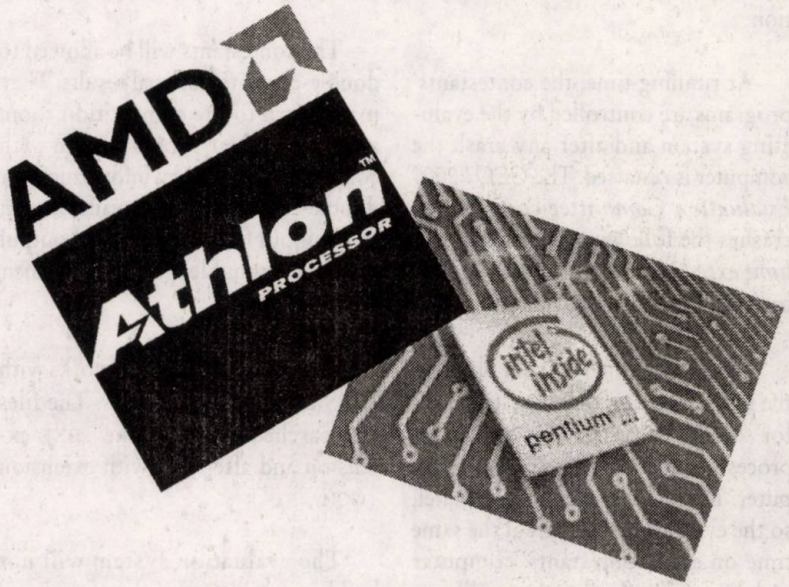
Intel's 820 and 840 chipsets could also use SDRAM but with a special adapter that translates the *Rambus* protocol back into SDRAM protocol, but does this with a significant performance hit, so systems using this combination are slower and more expensive than older chipsets that use SDRAM natively.

Soon after people found out that *Intel* wasn't backing *Rambus* just because they appreciated the technical capabilities of the new memory standard. *Rambus* granted *Intel* the right to buy 1 million shares at the nominal price of 1$ per action if *Intel* met certain milestones related to a specific percentage of chipsets that had to support *Rambus* memory. *Rambus* shares reached hights of 300$ on the stock exchange market, so the deal was obviously worth hundreds of millions to *Intel*, and could be worth even more if *Rambus* became the predominant memory standard.

But *Rambus* failed to take off as expected and *Intel* seem right now to make a u-turn and it is very likely that it will also support competing memory standards.

AMD, on the other hand, never expressed support for *Rambus*. It has remain loyal to the good old SDRAM and is publicly supporting a competing *Rambus* standard known as DDR (double data rate) as the memory of choice for its future processors.

Regardless of which memory type will be used, it is obvious that this will play a significant role in the future of the microprocessors. Unfortunately nowdays these decisions are made for strategic reasons rather than based on pure performance and cost issues.



Nothing New Under the Sun...

Evaluation at CEOI 2000

For most of the CEOI participants the evaluation method the organizers intend to use is not new. Why? It's simple. Many of them probably met a similar evaluation system in Turkey at the International Olympiad in 1999.

Like most national and international contests, the CEOI 2000 will use an automatic evaluation method for checking the contestants' solutions for the proposed tasks.

During the evaluation the contestants' presence is not required. This will save time for the contestants and the *Evaluation Committee*. It is a gain for all people involved. Moreover, the contestants will be spared of the emotions of assisting the evaluation of their own programs.

The contestants will have to build executable files for each of the tasks and place them into the specified directory. The evaluation program will locate these executables and provide input files for each step of the evaluation.

At running time, the contestants' programs are controlled by the evaluating system and after any crash the computer is restarted. The *CEOI 2000 Evaluation Committee* considers as crashes the following situations: *time limit exceded*, *hardware failure*, *timer violation* and *illegal operation*.

The evaluation will take place on the same computer the contestant used for solving the tasks. The evaluation process is supervised by a central computer. Everything is done in parallel, so the evaluation will start at the same time on each contestants' computer and, hopefully, finish very quickly.

This evaluation system is based on other's experience. At IOI'99 a similar system was used for evaluating the contestants' solutions.

During the whole process of evaluating and scoring the identification of the programs is done using the contestant's ID. Names and other information about the contestants are linked to their results only at the moment of printing of results sheets and lists. So, at the evaluation time, the *Evaluation Committee* does not know who was the contestant or who used a certain computer.

After the evaluation process each contestant will receive a detailed results sheet describing the behaviour of their programs.

The contestants will be allowed to double-check the official results. They may return to the competition room and re-test their programs. On each computer all necessary information for double-checking will be available, that is: the input files, the expected output files and output files generated by their programs.

The evaluation system works with DOS executable programs. The files are searched first with the `.EXE` extension and after that with extension `.COM`.

The evaluation system will not build executables using the contestants' source files because some of them may need special compiling options. Therefore it is the contestants' duty to build the executables using the desired compiling options.

The contestants must be very careful when naming and placing the source and executable files using the exact names specified in the task descriptions. Failure to do so will automatically lead to the contestant's obtaining no points for the corresponding task because the evaluation system will not be able to find the needed files. Also, it is very important for the programs to generate the output data to the specified output file which must be created in the current directory. Otherwise the evaluation system will not be able to locate the output files and will consider that these files were not generated. Another important aspect is to read input data from the specified input file. This file is always located in the current directory. Attempting to read data from other files or locations will make the contestant's programs run improperly.

Because there is a time limit for each test, the contestant's programs should not expect keyboard or mouse input. It is also advisable not to write anything to the standard output (screen) because this will only slow down the programs. The evaluation system will ignore all data written to the screen. Only the data written to the specified output file will be taken into consideration.

sponsors of CEOI 2000

MINOLTA

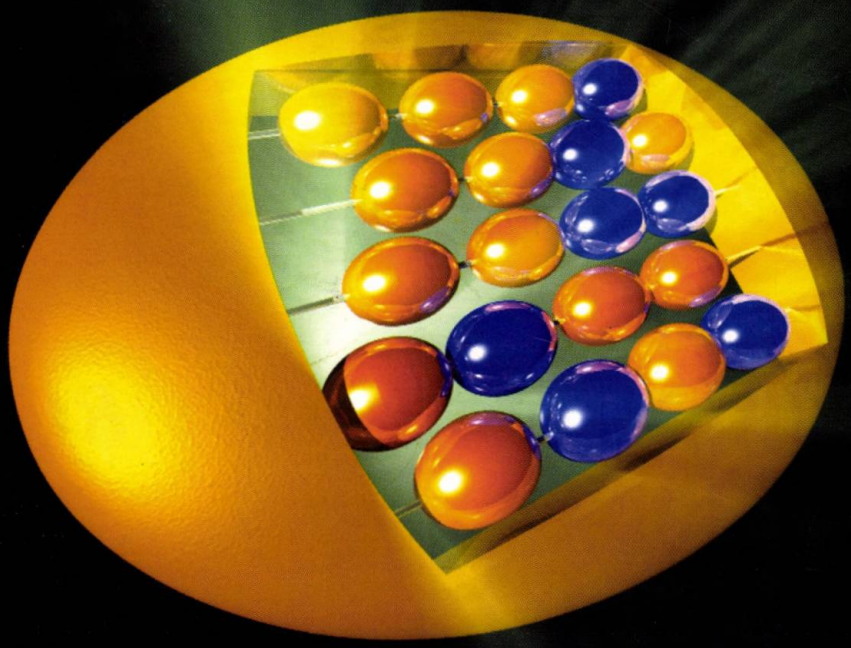
Ministry of National Education

7th Central European Olympiad in Informatics

CEOI 2000

CLUJ NAPOCA · ROMANIA

School Inspectorate-Cluj, Babeș-Bolyai University



Cluj Napoca August 24-31 2000

sponsors

A&C INTERNATIONAL S.A.

COMPUTER LIBRIS AGORA

COMPUTER PRESS AGORA

ARDAF

AUTODESK

BRINEL

CISCO SYSTEMS

COMPAQ

COMPACE COMPUTERS

ERICSSON

GENESYS IT Solutions - IT Infrastructures

FORTE COMPUTERS Flexybility. Made in Romania

GAZETA DE INFORMATICĂ GIFO

HEWLETT PACKARD

IBM

LOCKHEED MARTIN

MINOLTA

Microsoft

NEXTRA

ROMTELECOM S.A.

ROXER X

SCO

SOFTWIN SOFTWARE DEVELOPMENTS

SYSCO

TORNADO Sistems the connectivity company

turism transilvania

Xnet www.xnet.ro